Miscellaneous

- Use of this standard provides a convenient method for accessing CD-ROM, tape, hardfile, scanner and optical drives.

2.9.2 IDE

IDE is an optional interface for hardfiles. Requirements and recommendations follow:

Requirements

- Systems that are implemented with IDE must comply with the X3.221 AT Attachment (Revision 4.A); Proposed American National Standards.

Recommendations

- When the enhanced IDE interface becomes a standard and components are available, it is recommended
/ that systems which use IDE use this interface for improved data rates and that system software use the
/ capability for expanded capacity above 520 MB.

2.9.3 Ethernet

Requirements follow:

Requirements

- If Ethernet is implemented, it must adhere to the IEEE 802.3 standard. This specification covers both the electrical interface and the connectors.

2.9.4 Token Ring

Requirements follow:

Requirements

- If Token Ring is implemented, it must adhere to the IEEE 802.5 standard. This specification covers both the electrical interface and the connector.

2.9.5 Serial

Requirements, recommendations, and miscellaneous information follow:

Requirements

/ • The EIA/TIA-232-E standard for computer serial port connectors must be used for the required serial
/ port.

Recommendations

- It is strongly recommended that compliant systems implement EIA/TIA-232-E using a nine-pin D-shell male connector and pin assignments as defined below.
/ • It is strongly recommended that systems use the pin configuration and signal assignments for the
/ nine-pin serial port as shown in the following figure and table (viewed from the back of the machine).

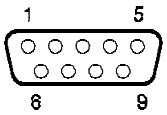

Figure 1. Nine-Pin Serial Connector Pin Arrangement

Table 2. Nine Serial Port Connector Signal and Pin Assignments	
Pin No.	**Signal Name**
1	Data Carrier Detect
2	Receive Data
3	Transmit Data
4	Data Terminal Ready
5	Signal Ground
6	Data Set Ready
7	Request To Send
8	Clear To Send
9	Ring Indicator

Miscellaneous

- The voltage levels are EIA only. A current loop interface is not supported.

2.9.6 LocalTalk

LocalTalk is the standard Macintosh serial port. Recommendations follow:

Recommendations

- It is strongly recommended that compliant systems implement EIA-422-A using the nine-pin connector and the following pinout.

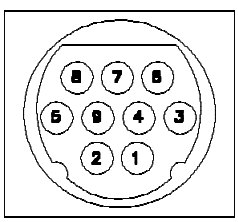

Figure 2. Connector -- Nine-Pin Arrangement

Table 3. Mini Nine-Pin Serial Port Connector Signal and Pin Assignments	
Pin No.	**Signal Name**
1	$SCLK_{out}$ Reset pod or get pod attention
2	$Sync_{in}/SCLK_{in}$ Serial clock from pod (up to 920 Kbit/sec)
3	TxD- Transmitted data inverted
4	Gnd/shield Signal Ground
5	RxD- Receive data inverted
6	TxD+ Transmitted data
7	Wakeup/TxHS Wakeup CPU or do DMA handshake
8	RxD+ Receive data
9	+5 V Power to pod (350 mA maximum)

2.9.7 Parallel Port Capability

Requirements and miscellaneous information follow:

Requirements

- If a system includes parallel ports, then those ports must be compliant to the standard specified by IEEE P1284, *Standard Signaling Method for a Bi-directional Parallel Peripheral Interface for Personal Computers.*

Recommendations

- It is strongly recommended that parallel ports of a system support the Extended Capabilities Port (ECP) in addition to the Compatibility Mode.

Miscellaneous

- P1284 defines "compliant" as the "Compatibility Mode" and the "Nibble Mode." The "Compatibility Mode" is backward compatible with many existing devices, including the PC parallel port. The "Nibble Mode" provides a bi-directional operations capability. The Extended Capabilities Port Mode (ECP) provides additional capabilities over this compatible mode.
- The Centronics interface is the popular name for the parallel printer port used by most "IBM compatible" personal computers. This interface has never been formalized.

2.9.8 PCI Bus

Miscellaneous information follows:

Miscellaneous

- PCI documents are available from and maintained by the PCI Special Interest Group. The PCI architecture is described in the following documents:
  - *PCI Local Bus Specification*, Revision 2.0, April 30, 1993
  - *PCI System Design Guide*, Revision 1.0, September 1993
  - *PCI to PCI Bridge Architecture Specification*, Revision 1.0, April 5,1994
  - *PCI Multimedia Design Guide*, Revision 1.0, March 29, 1994

2.9.9 PCMCIA Bus

Requirements and miscellaneous information follow:

The PCMCIA standard defines the physical requirements, electrical specifications and software architecture for the 68-pin cards and their sockets. Requirements, recommendations and miscellaneous information follow:

Requirements

- If a system has PCMCIA, then it must support the sockets that are release 2.x compatible.
- For maximum compatibility and interoperability, the system platform vendors must provide Socket Services and the operating system vendors must provide the Card Services extension.
- Both Socket Services and Card Services must be provided in the system abstraction software.
- The PC card vendors must provide Card Enablers (i.e. installers) which are clients of Card Services.
- If the operating system supports Plug and Play, then the Configuration Manager must take over individual Card Enablers.
- In addition to any default drivers, Card Services must allow for the use of Memory Technology Drivers to support additional types of memory devices.

Recommendations

- It is strongly recommended that the Card Services provide a default Memory Technology Driver.

Miscellaneous

- The PCMCIA software architecture has two key elements: Socket Services and Card Services. Socket Services is a hardware-dependent interface. The purpose of Socket Services is to mask the socket's actual hardware implementation from higher-level software components that utilize it. Card Services is a software layer that sits above Socket Services, coordinating access among the cards, the sockets and system resources such as interrupts and the memory map. Card Services accesses cards via Socket Services. The card drivers interact with the card via Card Services. In general, Card Services is operating system dependent.
- The PCMCIA standards can be ordered from and are maintained by the Personal Computer Memory Card International Association. The PCMCIA standards include the following individual specifications:
  - *PC Card Standard Specification*, release 2.1, July 1993
  - *Socket Services Specification*, release 2.1, July 1993
  - *Card Services Specification*, release 2.1, July 1993
  - *PC Card ATA Mass Storage Specification*, release 1.02, July 1993
  - *AIMS Specification*, release 1.01, November 1992
  - *Recommended Extensions*, release 1.00, November 1992
- In the future when the PCMCIA Card Bus Specification is approved, Card Bus should be considered if power consumption is within the allowable range.

2.9.10 ISA Bus

Requirements, recommendations, and miscellaneous information follow:

Requirements

- Systems must comply with the IEEE definition of ISA when it is approved.
- If Plug and Play is implemented on a system, then each Plug and Play hardware device must be able to be uniquely identified, must state the services it provides and the resources that it requires, must identify the driver that supports it, and must allow software to configure it.

Recommendations

- When the ISA Plug and Play standard is fully defined and accepted, it is strongly recommended that systems comply with the Plug and Play specifications. The Plug and Play specifications define hardware or software protocols to let systems perform configuration automatically.

Miscellaneous

- When the ISA bus was originally implemented, there was no architectural definition for the IBM Personal Computer AT Bus, only the implementation defined by the schematics published in the Technical Reference Manual. Since that time there have been several definitions of ISA put forward, including, but not limited to, the Family 1 Bus Architecture, an IEEE standards group, and an Intel publication.
- The IEEE definition of the ISA bus is in draft form and is called IEEE 996, *A standard for an Extended Personal Computer Back Plane Bus*.

2.9.11 Input Device Interfaces

Miscellaneous information follows:

Miscellaneous

- Examples of the interface for the alphanumeric and pointing devices include either ADB standard as used in Apple computers, or the PC/AT-PS/2* interface as implemented in an Intel 8042AH chip. The ADB interface circuit is described in the *Guide to Macintosh Family Hardware*.

2.10 System Configurations

Table 4 defines the required subsystem components for the minimum PowerPC Reference Platform-compliant system and recommended components for five typical computer configurations. Within this table, fields are marked as "None" to indicate that the device is not recommended, "Optional" to indicate that the hardware system vendor has the option to include the device, and "Required" to indicate that the device is required in all hardware configurations. The five configurations are:

Portable System	The portable configuration specifies the subsystems recommended for a PowerPC Reference Platform-compliant portable. A portable is a machine that is capable of battery operation.
Medialess System	The medialess system relies on network connections for all storage. Boot is from the network; software, data and paging space are attained from the network.
Desktop System	The desktop system is an entry-level system for commercial or technical applications.
Technical Workstation System	The technical workstation configuration specifies a technical user's desktop or deskside machine.
Server System	The server configuration specifies a machine that serves multiple users and does not require a locally attached keyboard and graphics display. Examples of functions performed by servers include backup server, compute server, data server, or print server.

Table 5 lists the minimum hardware configuration requirements for operating systems to run on a PowerPC Reference Platform hardware system. This table gives the requirements for the smallest version of the operating system required to support a stand-alone desktop workstation. Appendices for each operating system list the requirements for other configurations and list optional expansion and recommended features. The first column in Table 5 lists the minimum hardware configuration required to support any one of the oper-

ating systems. The remaining columns list the minimum hardware configuration requirements for each operating system.

Hardware system vendors must use the operating system information and the required minimum hardware configuration information in this section to configure a system. The hardware system vendor could use this information to tailor an implementation to include some operating systems in some configurations while excluding others, or could configure a machine large enough to hold any one or several of the operating systems.

Table 4 (Page 1 of 2). Subsystem Components of PowerPC Reference Platform Systems

Subsystem	Required Minimum	Recommended Typical Configurations				
		Portable	**Medialess**	**Desktop**	**Technical**	**Server**
Processor	PowerPC	PowerPC	PowerPC	PowerPC	PowerPC	PowerPC
System Memory	8 MB	Expandable to 32 MB (min.), parity	Expandable to 32 MB (min.), parity	Expandable to 32 MB (min.), parity	Expandable to 32 MB (min.), ECC	Expandable to 32 MB (min.), ECC
System ROM	Sized as Needed	Sized as Needed, Software Write-able	Sized as Needed, Software Write-able	Sized as Needed, Software Write-able	Sized as Needed, Software Write-able	Sized as Needed, Software Write-able
Non-volatile Memory	4 KB	4 KB	4 KB	4 KB	4 KB	4 KB
L2 Cache	Optional	Optional	Optional	Optional	> = 256 KB	> = 256 KB
Hardfile	120 MB*	240 MB	None	240 MB	240 MB	400 MB
Floppy	Optional**	1.44 MB MFM	None	1.44 MB MFM	1.44 MB MFM	1.44 MB MFM
CD-ROM	Optional**	Optional	None	XA, >=300 KB/s	XA, >=300 KB/s	XA, >=300 KB/s
Alphanumeric Input Device	Required	Keyboard	Keyboard	Keyboard	Keyboard	Keyboard or Terminal
Pointing Device	Required with Keyboard	Required	2-Button Mouse	2-Button Mouse	2-Button Mouse	Required with Keyboard
Audio	16-Bit Stereo, 22.05 and 44.1 KHz	Multi-voice, Compression	Multi-voice, Compression	Multi-voice, Compression	Multi-voice, Compression	16-Bit Stereo, 22.05 and 44.1 KHz
Graphics	640x480x8	640x480x8	>=1024x768x16	>=1024x768x16	>=1024x768x16	Possibly Terminal
Real-Time Clock	Required	Required	Required	Required	Required	Required
Serial Port	EIA/TIA-232-E at 19.2K	> = 19.2K	> = 19.2K	> = 19.2K	> = 19.2K	> = 19.2K

Table 4 (Page 2 of 2). Subsystem Components of PowerPC Reference Platform Systems

Subsystem	Required Minimum	Recommended Typical Configurations				
		Portable	**Medialess**	**Desktop**	**Technical**	**Server**
Parallel Port	Optional	P1284 ECP Mode	P1284 ECP Mode	P1284 ECP Mode	P1284 ECP Mode	P1284 ECP Mode
Network	Optional**	Ethernet or LocalTalk	Ethernet or LocalTalk	Ethernet or LocalTalk	Ethernet or LocalTalk	Ethernet or LocalTalk
Expansion Bus(es)	Optional	PCI, ISA, or PCMCIA	PCI, ISA, or PCMCIA	PCI, ISA, or PCMCIA	PCI, ISA, or PCMCIA	PCI, ISA, or PCMCIA
SCSI	Optional	Optional	Optional	SCSI-2 Fast	SCSI-2 Fast	SCSI-2 Fast

Note:

* Hardfile capability may be provided or augmented by a network connection, in which case a network connection is required.

** Some method of installing the operating system, applications and data is required.

Table 5 (Page 1 of 2). Operating System Requirements for Subsystem Components						
Subsystem	**Required to Support Any One**	**Minimum Required for Each Operating System**				
		Windows NT	**AIX**	**Workplace**	**Solaris**	**Taligent**
Processor	601, 603, 604	601, 603, 604	601, 603, 604	601, 603, 604	601, 603, 604	
System Memory	16 MB	16 MB	16 MB	8 MB	16 MB	
System ROM	Standard	Standard	Standard	Standard	Standard	
Non-volatile Memory	Standard 4 KB	Standard 4 KB	Standard 4 KB	Standard 4 KB	Standard 4 KB	
L2 Cache	Optional	Optional	Optional	Optional	Optional	
Hardfile	200 MB	200 MB	200 MB	120 MB	200 MB *	
Floppy (1.44 MB MFM)	Required	Optional	Optional	Required *	Required	
CD-ROM *	Required	Required	Required	Required	Required	
Alphanumeric Input Device	Required	Required	Required	Required	Required	
Pointing Device	Required	Required	Required	Required	Required	
Audio	Standard	Standard	Standard	Standard	Standard	
Graphics	800x600	640x480	800x600	640x480	640x480	
Real-Time Clock	Standard	Standard	Standard	Standard	Standard	
Serial Port	2 Required	Standard	2 Required	Standard	Standard	
Parallel Port	1 Required	Optional	1 Required	Optional	Optional	
Network	Optional	Optional	Optional	Optional	Optional	
Expansion Bus(es) **	ISA, PCI	ISA, PCI	ISA, PCI	ISA, PCI, PCMCIA	ISA, PCI, PCMCIA	

Table 5 (Page 2 of 2). Operating System Requirements for Subsystem Components

Subsystem	Required to Support Any One	Minimum Required for Each Operating System				
		Windows NT	AIX	Workplace	Solaris	Taligent
SCSI **	1 Required	Optional	1 Required	Optional	Optional	

Legend:

Entry	Definition
Standard	indicates that the component is required in all hardware configurations
Required	indicates that this component must be present to support that particular operating system
Optional	indicates that the operating system supports the component and that the hardware system vendor has the option to include the component
*	Capability can be provided via a network.
**	Currently supported devices. Different devices could be accommodated by providing alternative abstraction software.

3.0 Architecture Guidance

This section defines the collection of architectural constraints and conventions for PowerPC Reference Platform-compliant systems. Some features described within this section are required to support any operating system. These features are stated in terms of "must." Every PowerPC Reference Platform-compliant machine must have these features. Some features are recommended for better usability or performance. These features are described in terms of "recommended." Some possible features are discouraged for future
/ compatibility reasons and for hardware simplification reasons. In some cases, additional information is pre-
/ sented to help explain the implementation of these requirements and recommendations. This section is written primarily for hardware system vendors, but there are some software requirements and recommendations.

3.1 System Topology and Coherence

Miscellaneous information follows:

Miscellaneous

- A typical system topology used by PowerPC Reference Platform systems is shown in Figure 3. All PowerPC Reference Platform systems consist of one or more compliant processors; a volatile memory separate from other subsystems used for storage of data and instructions, called System Memory; and a number of objects, called I/O subsystems, that may initiate transactions to System Memory. The processors are linked over the *primary processor bus* to each other, to System Memory, and to a bus bridge. In general, I/O devices do not connect to the PowerPC processor bus. The bus bridge connects to a *secondary bus* which has I/O subsystems connected to it. In turn, another bus bridge may be employed to a *tertiary bus* with additional I/O subsystems connected to it. Typically the bus speeds and throughput decrease and the number of supportable loads increases as one progresses from the primary processor bus to the tertiary bus. PowerPC Reference Platform multiprocessor systems have a symmetric (at least from the hardware point of view) shared memory model. Please refer to Section 3.13, "Multiprocessor Considerations," for architecture considerations specific to multiprocessor systems.
- There are variations on this typical topology which are likely to occur and are therefore worth
/ describing. The secondary bus may be implemented as two or more parallel expansion buses for performance reasons. Similarly the tertiary bus may be two or more parallel expansion buses. The bus bridge and/or memory controller may be integrated into the processor chip. In the case where the bus bridge is integrated into the processor chip, the primary processor bus would not be the PowerPC processor bus, but would be an expansion bus (e.g. PCI). I/O subsystems would normally be attached to the primary processor bus in this case. If this bus is the only bus coming from the processor chip, then the System Memory would continue to be attached to the primary processor bus. If the memory controller was integrated into the processor chip, then System Memory would be attached to the processor chip.

3.2 System Memory

Requirements and miscellaneous information follow:

Requirements

- Access to System Memory by processors and I/O must conform to the coherency requirements specified by *The PowerPC Architecture*, when coherency is required.

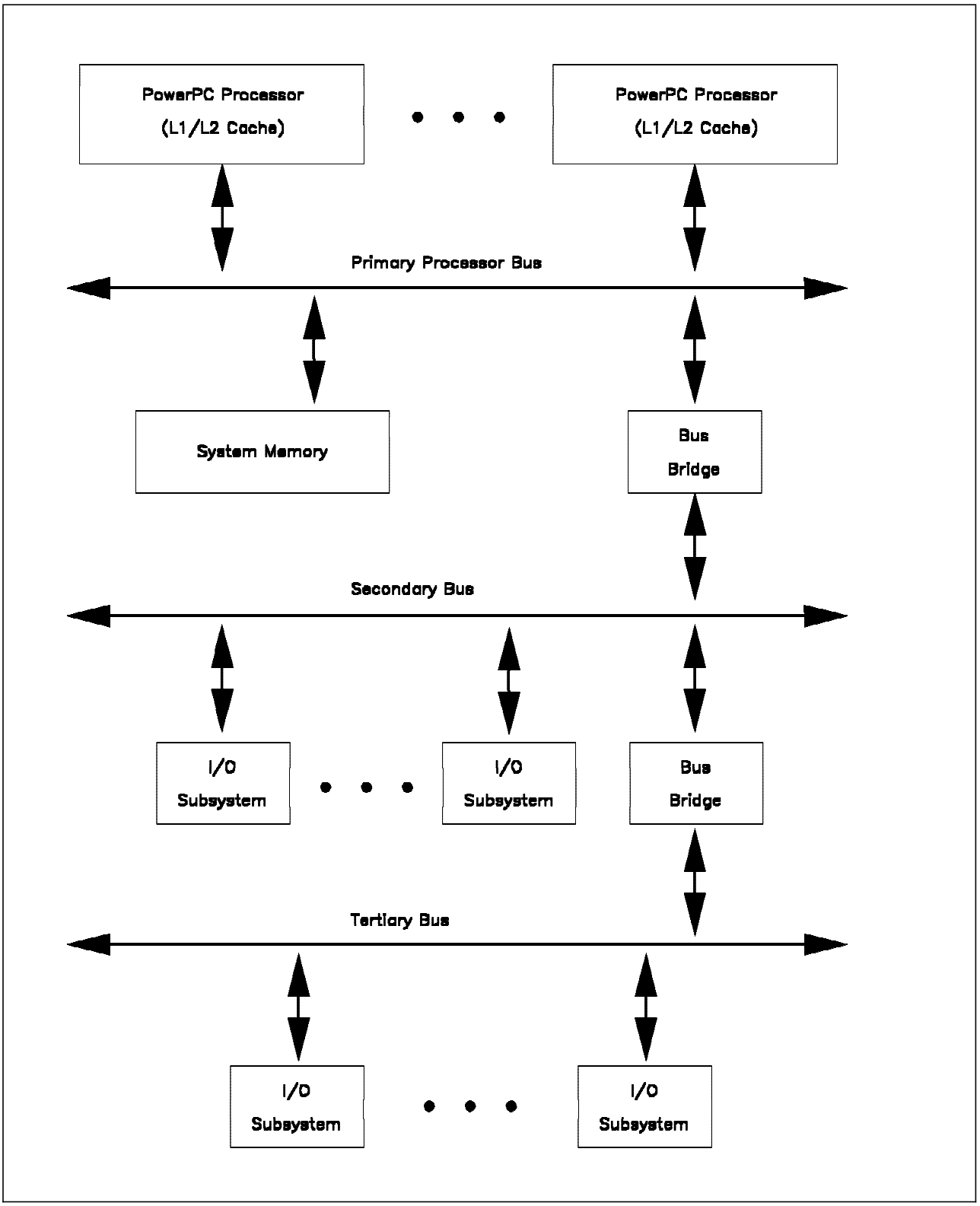

Figure 3. Typical PowerPC Reference Platform System Topology

- For coherence required pages, a coherent view of storage must be maintained between all processor caches and System Memory for any System Memory reference transactions initiated by the processor and for any transactions to System Memory initiated from an I/O bus. In particular:

- transactions between a processor and System Memory
- transactions between an I/O subsystem and System Memory
- transactions between any processor in a multiprocessor system and System Memory

Miscellaneous

- Coherence among any other storage elements or for any other kinds of transactions need not be maintained by hardware. In particular, transactions between an I/O device and I/O Memory need not be maintained coherent by the hardware.
- The PowerPC architecture allows software to set the coherence mode of System Memory as "memory coherence required" or "memory coherence not required." When memory coherence is required the hardware system maintains all data buffers in the L2 cache, in the memory controller, and in the bus bridge to the secondary bus in coherence.
- Some PowerPC processors may not assert the bus coherency tag bit (M bit) on instruction fetches; nevertheless, instruction fetches should be coherent with respect to prior write transactions between I/O devices and System Memory.
- Transactions between the I/O subsystem and System Memory are required to be coherent when the I/O operation is complete. An I/O operation is considered complete when the resulting I/O interruption processing is complete. In addition, most I/O bus architectures require strong ordering of I/O accesses to System Memory while an I/O operation is in process.
- Programmed I/O write transactions by a processor to an I/O device are required to be strongly ordered with respect to prior read transactions by that device to System Memory. Similarly, programmed I/O read transactions by a processor to an I/O device are required to be strongly ordered with respect to prior write transactions by that device to System Memory.
- Store combining (described in *The PowerPC Architecture*, Book III) by some PowerPC processors may have an effect on the order in which data is stored.

3.3 I/O Memory

PowerPC Reference Platform systems may contain CPU-visible read/write memory other than the System Memory. This memory is located on I/O devices which are normally attached to secondary or tertiary buses. These devices may be accessed through the bus bridge on the primary processor bus. Requirements and miscellaneous information follow:

Requirements

- I/O Memory which is not well behaved in terms of prefetching and other speculative storage operations must be marked as guarded.

Miscellaneous

- "Guarded" is a term used in *The PowerPC Architecture*, Book III. Storage which is not well behaved in terms of prefetching and other speculative storage operations must be marked as guarded. Examples of this type of storage include I/O subsystems, memory with holes, and the top of System Memory which has no successor. With a few exceptions, most of the I/O Memory would be mapped as guarded. Graphics frame buffers are an example of I/O Memory which would not be marked as guarded.
- The 601 treats all memory as not guarded in the sense that speculative fetching may occur to any area of memory. The 601 does speculative instruction execution. In this way, data which is adjacent to instructions could be speculatively executed. The 601 does not perform any speculative load or store instructions.

3.4 System I/O

Requirements, recommendations, and miscellaneous information follow:

Requirements

- Software must assure that processor implementations of load and store combining do not violate bus and device operational constraints.
- All transfers initiated by a processor, independent of the target of the transfer, must obey PowerPC architectural semantics for memory ordering (including semantics for *eieio* and *sync*).

Recommendations

- It is recommended that all loads or stores to System I/O match the device register size.
- If load and store combining by the processor would violate bus and device operational constraints, then it is recommended that software separate adjacent loads and stores with storage synchronizing instructions (e.g. *eieio*).

Miscellaneous

- The PowerPC architecture allows processors to implement "load or store combining." This architectural feature allows a processor to combine one or more loads or stores to adjacent storage into a single bus transaction. In this way, two store word instructions to adjacent locations could become a doubleword bus transfer.

3.5 Self-Modifying Code

Programs which modify the currently executing sequence of instructions, self-modifying code, are allowed on PowerPC Reference Platform systems. Requirements, recommendations, and miscellaneous information follow:

Requirements

- Self-modifying code must be done in such a manner that the instruction fetch pipeline and cache do not contain the original form of the modified code.

Recommendations

- It is recommended that operating system software provide a service which will allow applications that modify code to ensure that instructions and data in cache and System Memory are coherent.
- The recommended instructions for obtaining instruction and data coherence are as follows:
  a) *dcbst* -- update System Memory
  b) *sync* -- wait for update
  c) *icbi* -- remove (invalidate) copy in instruction cache
  d) *sync* -- wait for *icbi* to be globally performed (may be omitted in uniprocessor systems)
  e) *isync* -- remove copy in own instruction buffer
- Because of the overhead incurred while performing the instruction and data storage synchronization, it is recommended that frequent minor self-modification to the instruction stream be avoided.

Miscellaneous

- The PowerPC architecture does not enforce consistency between the instruction cache and System Memory.

3.6 Resource Locking

The *PowerPC Reference Platform Specification* does not require resource locking capability beyond the atomic access features provided in the PowerPC architecture. Some buses provide a resource locking function (e.g. PCI exclusive access mechanism) for use by bus masters. System designs may choose not to implement resource locking functions. If resource locking is provided, then the following requirements, recommendations, and miscellaneous information apply:

Requirements

- If resource locking is provided for bus masters, then systems must not allow the entire bus to be locked.
- Locking of a system memory resource by bus masters is acceptable, but memory system coherency must always be maintained.
- Bus bridges which support the locking of system memory between devices and device drivers or other processor tasks must present a read-with-intent-to-modify transaction to the processor interface when a device attempts to lock a system memory resource.
- Locking of system memory resources by bus masters must occur in increments of less than or equal to a memory system page (4 KB), and must not cross a page boundary.
- Processor operations which require the use of locking features must use the PowerPC atomic access feature (e.g. *lwarx* and *stwcx*).

Recommendations

- Locking of system memory in cache line increments and aligned on cache line boundaries is recommended.
- If resource locking is implemented for a given bus architecture, then it is recommended that the implementation fully support the resource locking specification for that architecture.
- Other than the atomic access features provided by PowerPC processors, additional hardware support for locking operations may vary among systems. It is recommended that devices and device drivers be designed to work in the absence of such support.

Miscellaneous

- Requiring a read-with-intent-to-modify transaction to be presented to the processor interface when locking a system memory resource between a device and its device driver causes cached elements affected by the lock to be flushed and invalidated before the lock is granted. It will also cause reservations to a cached element affected by the lock (e.g. those obtained by a processor via a *lwarx* instruction) to be cancelled. Subsequent attempts to access the resource, except by the lock owner, can then be retried while the lock is valid.
- Lock operations will generally be of three types:

  a) locking of a slave device by a master

  b) locking a system memory resource between two masters

  c) locking a system memory resource between a master and its device driver

  It is likely that the third type will be the most widely used by devices.

3.7 Bus Errors and Unsupported Bus Transactions

Requirements and miscellaneous information follow:

Requirements

- If bus errors (e.g. detected parity errors) and unsupported transaction types (e.g. I/O of an unaligned 4-byte word) to local and expansion buses are to be reported, then the system must send a high-priority interrupt (e.g. soft reset, NMI, INT0, or Transaction Error Acknowledgement -- TEA) to the PowerPC processor.

Miscellaneous

- The level of error reporting in PowerPC Reference Platform-compliant systems is not defined in this specification, but is left up to the system designer. A minimal-cost implementation may choose to ignore error conditions and continue processing. A more robust design may detect and report error occurrences to software.
- When a PowerPC processor receives a TEA, it produces a "machine check" interrupt. The PowerPC architecture allows the implementation to define whether interrupts are "precise" or "imprecise." The PowerPC 601, 603, and 604 have defined the soft reset and machine check interrupts as imprecise, which means that the exact instruction address that caused the interrupt may not be reported to the interrupt handling software. Recovery from imprecise interrupts may be difficult or impossible. In most cases the interrupt handler may have to record the error conditions in NVRAM and stop the system.

3.8 Memory Map

Requirements, recommendations, and miscellaneous information follow:

Requirements

- The memory map must consist of four distinct areas, including:

System Memory	Used to store the programs and data being operated on by the processor
System I/O space	Used for input and output to devices attached to the I/O buses
I/O Memory	Memory on I/O devices such as local memory on graphics adaptors
System ROM	Contains the initial bring-up code

- The size and location of these memory spaces must be passed to the operating system loader in the residual data.
- System Memory (as real memory) space must start at 0.
- If the IP bit of the MSR is 0, then the area of System Memory from 0 through X'00002FFF' must be reserved for the interrupt vector table as defined by the PowerPC architecture.
- If the IP bit of the MSR is set to 1, then the area of the memory space from X'FFF0 0000' to X'FFF0 2FFF' (X'FFFF FFFF FFF0 0000' to X'FFFF FFFF FFF0 2FFF' in 64-bit processors) must be reserved for the interrupt vector table as defined by the PowerPC architecture.
- System ROM must be located at the top of the memory space because the processors begin execution after power is turned on at X'FFF00100' (X'FFFFFFFFFFF00100' in 64-bit processors).

Recommendations

- The following describes the recommended approach that the boot process would use to pass the memory map to the operating system. The firmware and system registers implemented by the supporting hardware would contain enough information about the memory map to allow the bring-up process to construct the memory map. The bring-up process would extract this information, discover any holes in memory, and describe the memory map in the residual data. Section 5.6.1, "Map of Residual Data Structure," shows the residual data map which contains the structure "MEM_MAP" that is used to pass the memory map to the operating system. This abstraction of the memory map will allow variations from implementation to implementation and will provide for expansion of addressing to 64 bits.
- It is strongly recommended that memory map implementations accommodate the dedicated real memory required by AIX as described in Appendix E.7, "Boot Time Abstraction Requirements."

Miscellaneous

- Two example address maps are described below to demonstrate possible arrangements. These particular arrangements are not required by the *PowerPC Reference Platform Specification*.

3.8.1 Example Memory Maps

Figure 4 shows one variation of a PowerPC Reference Platform System Memory map. The left side of this figure shows the view of memory from the PowerPC processor. The right side of this figure shows the view
/ of memory of the I/O master doing I/O addressing or memory addressing. The large arrows in the figure
/ show correspondence of areas and not data flow which, in some cases, may be in only one direction. As shown on the left side of the figure, the address space is split into four areas: a System Memory portion with addresses from 0 to 2 GB, a System I/O portion which stretches from 2 GB to 3 GB, an I/O Memory area which covers 3 GB to 4 GB - 16 MB, and a System ROM and Register space from 4 GB - 16 MB through 4 GB.

The System Memory is addressed by the operating system and user applications executing in the PowerPC processor. Access to System Memory by the PowerPC processor is in the 0 to 2 GB range. System Memory accessed by an I/O master will be addressed in the 2 to 4 GB range as shown on the right side of the figure.

The System I/O address space is used for input and output to the I/O subsystems attached to the secondary or tertiary I/O buses (e.g. PCI, VL, ISA, or Micro Channel).

The I/O Memory area is used for graphics memory and other I/O-based memory.

The System ROM and space for system registers is allocated to the top 16 MB of the memory map. The System ROM contains the code for power-on self tests (POST), code for establishing the initial configuration of the system, code for bring-up, vital product data (VPD), and system-specific information.

As shown on the right side of the figure, the I/O master has two modes of addressing. The I/O master doing memory address mode transfers will see System Memory as having addresses from 2 GB to 4 GB. I/O masters addressing I/O Memory on adaptors such as VRAM will address this I/O Memory in the range of 0 to 1 GB. The upper 16 MB of this space is reserved for the System ROM and system registers and is not addressable by the I/O master. In the memory-addressing mode, the I/O master cannot address the space from 1 GB to 2 GB. The I/O master doing I/O addressing mode transfers will see the System I/O space as having addresses from 0 to 1 GB.

Within this architecture, treatment of addresses outside of the allowed ranges is implementation dependent.

Figure 5 shows a different PowerPC Reference Platform memory map. The first 640 KB contain intra-system communication information such as the stack, scratch pad, interrupt vectors, residual data from the bring-up process, and interprocess messaging areas. The space between 640 KB and 1 GB is used for I/O Memory such as graphics buffers, cache for storage devices, network buffers and video buffers. System Memory runs from 1 GB to 3 GB. Memory space for System I/O runs from 3 GB to 4 GB - 16 MB. This area is used for memory-mapped I/O to devices on the I/O buses. System ROM and system registers occupy the space from 4 GB - 16 MB to 4 GB.

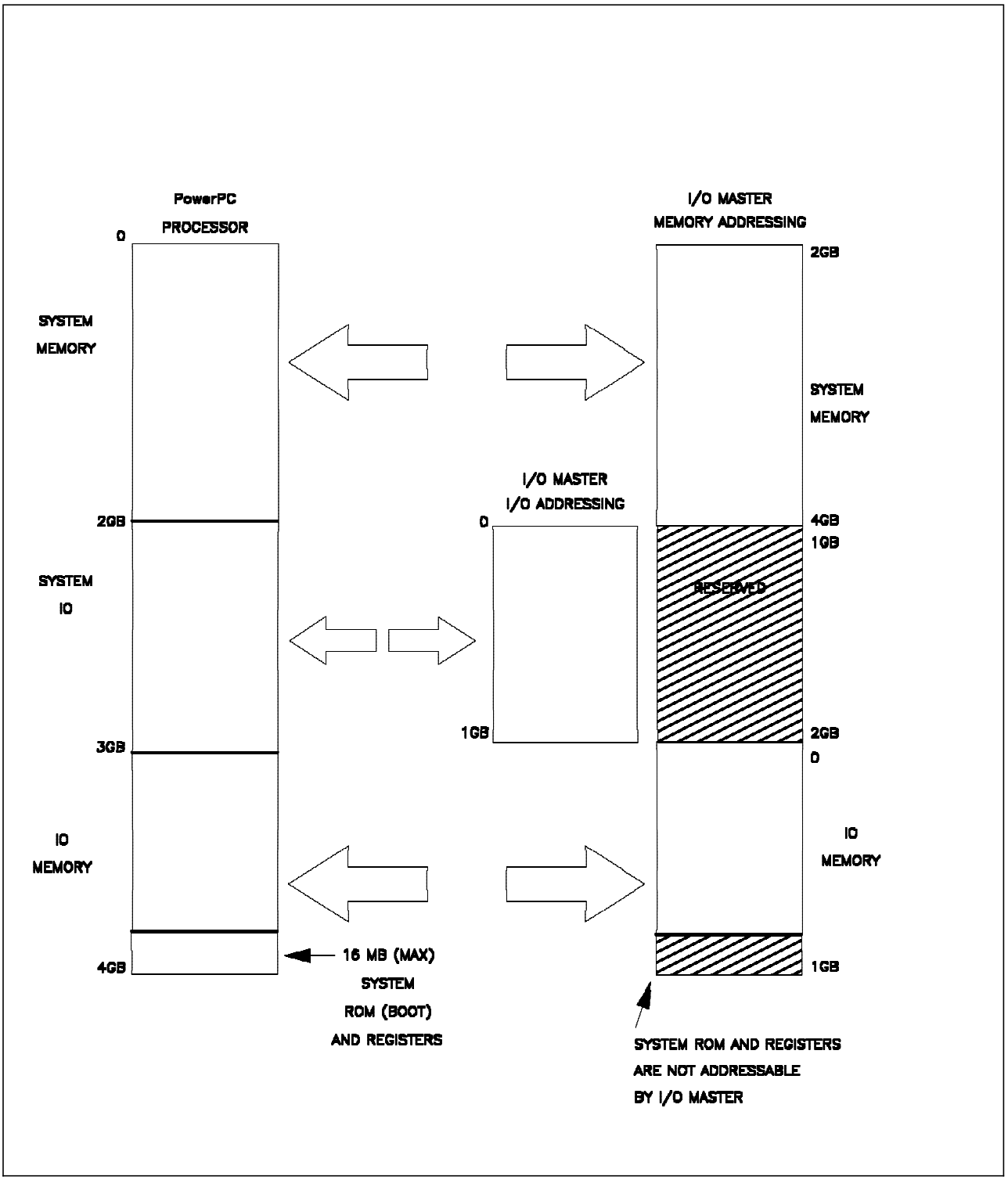

Figure 4. Example of a Memory Map -- Alternative 1

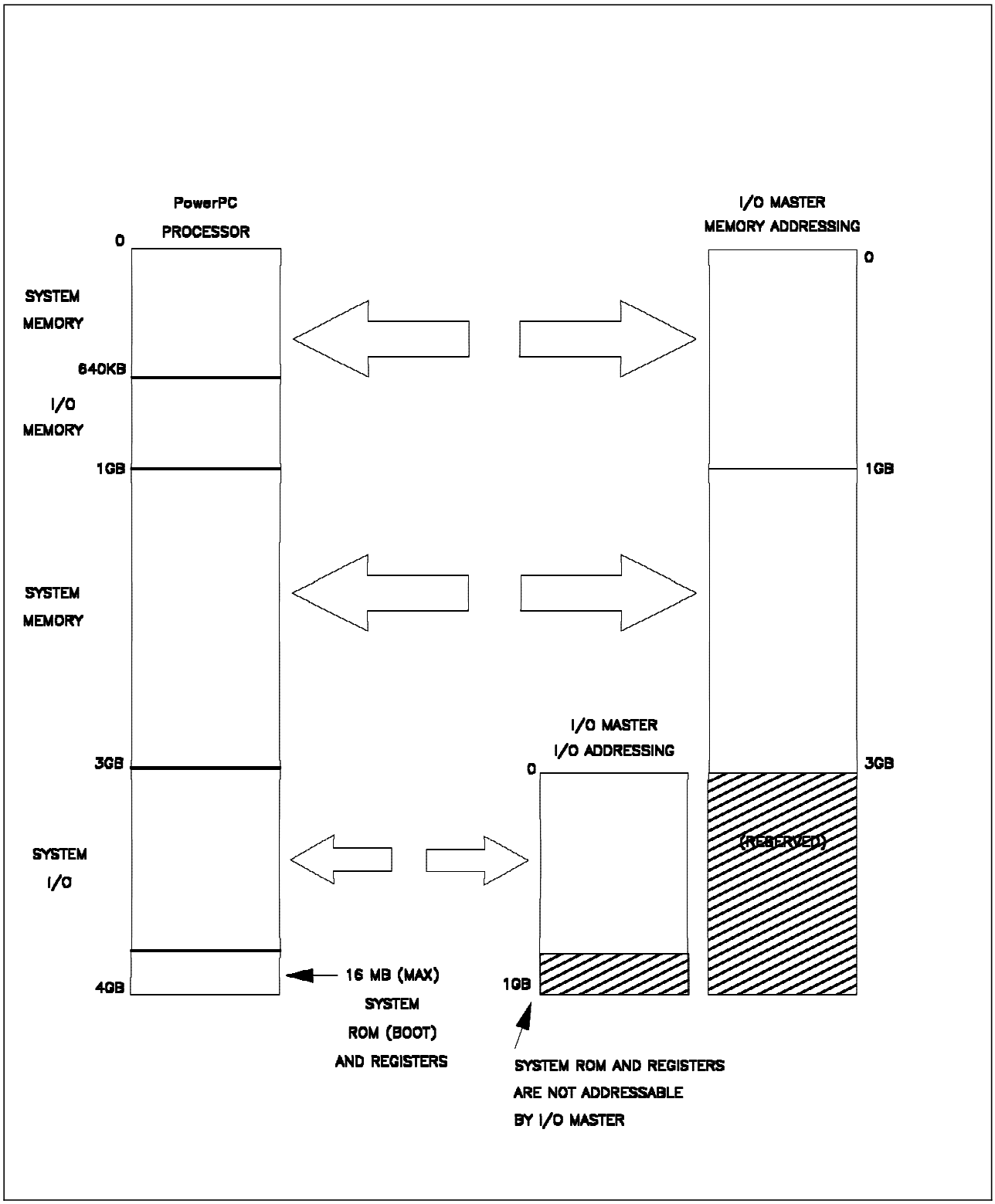

Figure 5. Example of a Memory Map -- Alternative 2

3.9 Memory Ordering

Requirements and miscellaneous information follow:

Requirements

/ • Since some processors support only a weakly ordered memory model, software must program for that
/ case.
• All transfers initiated by a processor, independent of the target of the transfer, must obey PowerPC architectural semantics for memory ordering (including semantics for *eieio* and *sync*).

Miscellaneous

- A machine conforms to a weakly ordered memory model if:
  - accesses to global synchronizing variables are sequentially consistent, and
  - no access to a synchronizing variable is issued in a processor before all previous global data accesses have been performed, and
  - no access to global data is issued by a processor before a previous access to a synchronizing variable has been performed
- "Sequentially consistent" as used above means that a load or store for any execution is the same as if the operations of all the processors were executed in some sequential order, and the operations of each individual processor appear in this sequence in the order specified by its program. A store is completed when the value stored by the processor executing the instruction can be seen by all other processors. A load is completed when the value to be returned by the load has been set and cannot be changed.
- Thus for a weakly ordered memory model, the order in which a processor performs storage accesses, the order in which those accesses complete in main storage, and the order in which those accesses are seen as completing by another processor may all be different.
- Storage accesses by a single processor without explicit synchronization operations may not appear to complete in program order when viewed from another processor. However, accesses to the same address from the same processor will complete in program order.
- The instructions *eieio* and *sync* may be used to enforce ordered storage access with respect to a given processor. Higher-level protocols (using *lwarx* and *stwcx* for example) must be used to ensure ordering between processors or between processors and devices.
- The operations *lwarx* and *stwcx* are defined only on effective addresses that are mapped to System Memory on the local PowerPC processor bus. (Note that PowerPC architecture does not specify that *lwarx* or *stwcx* necessarily generate externally visible transaction. Consequently, these restrictions on *lwarx* and *stwcx* are not enforceable in hardware.)
- For additional reading on memory ordering, refer to *Memory Access Buffering in Multiprocessors* by Michel Dubois, Christoph Scheurich, and Faye Briggs, in 13th ISCA, pages 434-441, 1986.

3.10 Configuration and Diagnostics

Requirements, recommendations, and miscellaneous information follow:

Requirements

- A PowerPC Reference Platform system must include some form of power-on self test.
/ • In addition, stand-alone diagnostics must be supplied for systems.

Recommendations

- It is recommended that stand-alone diagnostics be operating system independent.
- It is strongly recommended that each operating system support on-line configuration and concurrent maintenance (on-line diagnostics).

Miscellaneous

/ • While power-on self test and stand-alone diagnostics are sufficient for some customers' needs, many
/ others may require on-line configuration and diagnostics. On-line configuration and diagnostics are indicated by some of the likely attributes of near-term computer systems including:
  – increased power management
  – hot-plug capability
  – continuous availability
• A highly power-managed system is not often shut off or restarted. Thus the initial hardware checkout, test, and verification (e.g. power-on self test) and stand-alone diagnostics cannot realistically be relied upon to provide the primary means for system maintenance and diagnostics. Properly, this task should be part of the run-time environment supplied by an operating system.
• Hot-plug allows a hardware subsystem to be dynamically coupled or decoupled from the system. This will require the run-time environment supplied by the operating system to configure and verify the operation of any newly coupled device. It is likely that this function would be included in the respective device drivers. This function also complements the on-line diagnostics function. The result of diagnosis might be removal, replacement, and configuration of some I/O subsystem while the system is on-line.
• Continuously available systems are becoming more important. Workstations can fill the role of servers which could run continuously unattended. Workstations running as process monitors (e.g. ATM or process control) are other examples. In this environment, on-line maintenance and on-line configuration are both important attributes of the operating environment.
/ • Refer to Section A.7, "A Proposed Diagnostic Strategy," for a description of an implementation of diag-
/ nostics support.

3.11 Power Management

Two types of power management techniques have been used in the computer industry. The first type is hardware managed and is inaccessible to system software. This approach will be referred to as "micro power management." The second type uses system software to control hardware and will be referred to as "macro power management." Micro power management is the predominant technique in the existing base of PCs. This is due largely to the need for power management in portable computers before operating systems offered power management support. Macro power management is by far the more powerful technique and is thus the basis for the Reference Platform power management model.

Requirements, recommendations, and miscellaneous information follow:

Requirements

/ • If a hardware platform supports power management, then its power management features must be con-
/ trollable by system software.
/ • Abstraction software and device drivers must provide the necessary functions to support the power man-
/ agement features of its target operating system.

Recommendations

• It is strongly recommended that systems incorporate power management.
• It is recommended that system software and hardware work effectively together to provide the necessary functions to implement a macro power management model similar to that shown in Figure 6. In this model, the following is true:
  – The operating system is the power management control hub.
  – Power management policy is set by the user through an application interface.
  – The operating system implements power management policy through the system abstraction and device driver interfaces.
  – The operating system supports power-management-aware applications.

- It is recommended that operating systems abstract the following activities, and software abstraction services and device drivers support these activities:
  - Issuing power state change commands to the system, subsystems, and devices.
  - Monitoring devices and subsystems for power management event notifications and providing approval.
  - Receiving power management requests from devices and subsystems.

Miscellaneous

- Abstraction software and device drivers provide the connection between the operating system and hardware. This software is aware of hardware power management capabilities and translates operating system power management requests into specific device and subsystem controls. It also passes information provided by hardware to the operating system to help it perform power management.

3.11.1 Support for Suspend and Hibernation System States

This specification does not mandate particular system power states or algorithms for transition between states. Two states, however, will likely be implemented in most systems. These states require special consideration when designing hardware, abstraction software, device drivers, and operating systems. The first is a suspend state which is characterized by very low power dissipation and fast recovery to full function. Resumption from suspend generally occurs upon receiving an I/O interrupt resulting from mouse or keyboard activity, communications events, or when opening the lid of a laptop system. In most systems the contents of system memory will remain active while in the suspend state. Hibernation is the second state, and is defined as the system being turned off, with the system state being written to non-volatile storage such as hardfile. Recovery from hibernation would generally not be as quick as that from suspend, but it saves the time of loading the operating system and brings the system back to its state before it hibernated. Resumption from hibernation would occur when the system is powered back on.

Requirements, recommendations, and miscellaneous information follow:

Requirements

- Power-managed systems must provide the following infrastructure to support a suspend state:
  - Hardware must provide a software-writeable suspend flag. This bit can be held in NVRAM or in a register which maintains its value while the system is in the suspend state.
  - Hardware must provide a software-writeable resume pointer to indicate to firmware the correct starting point when resuming from suspend.
  - If there is a firmware requirement for a free contiguous memory space when resuming from suspend, then this must be communicated to system software.
- Compliant operating systems must abstract the location of the suspend flag and the resume pointer.
- Prior to suspending, the operating system must allocate the firmware requirement for free contiguous memory.
- If the hibernation state is implemented, the OS must set a hibernation flag prior to powering the system off to indicate to the OS loader that a saved system state exists.
- After resuming from hibernation or suspend, the operating system is responsible for updating its time of day using the system's persistent time-keeping mechanism.

Recommendations

- It is recommended that operating systems implement the system suspend and hibernate functions.
- It is recommended that firmware verify that the saved memory image is valid before jumping to the resume address when resuming from suspend.
- It is recommended that the saved system state is verified before loading it when the hibernation flag is set.

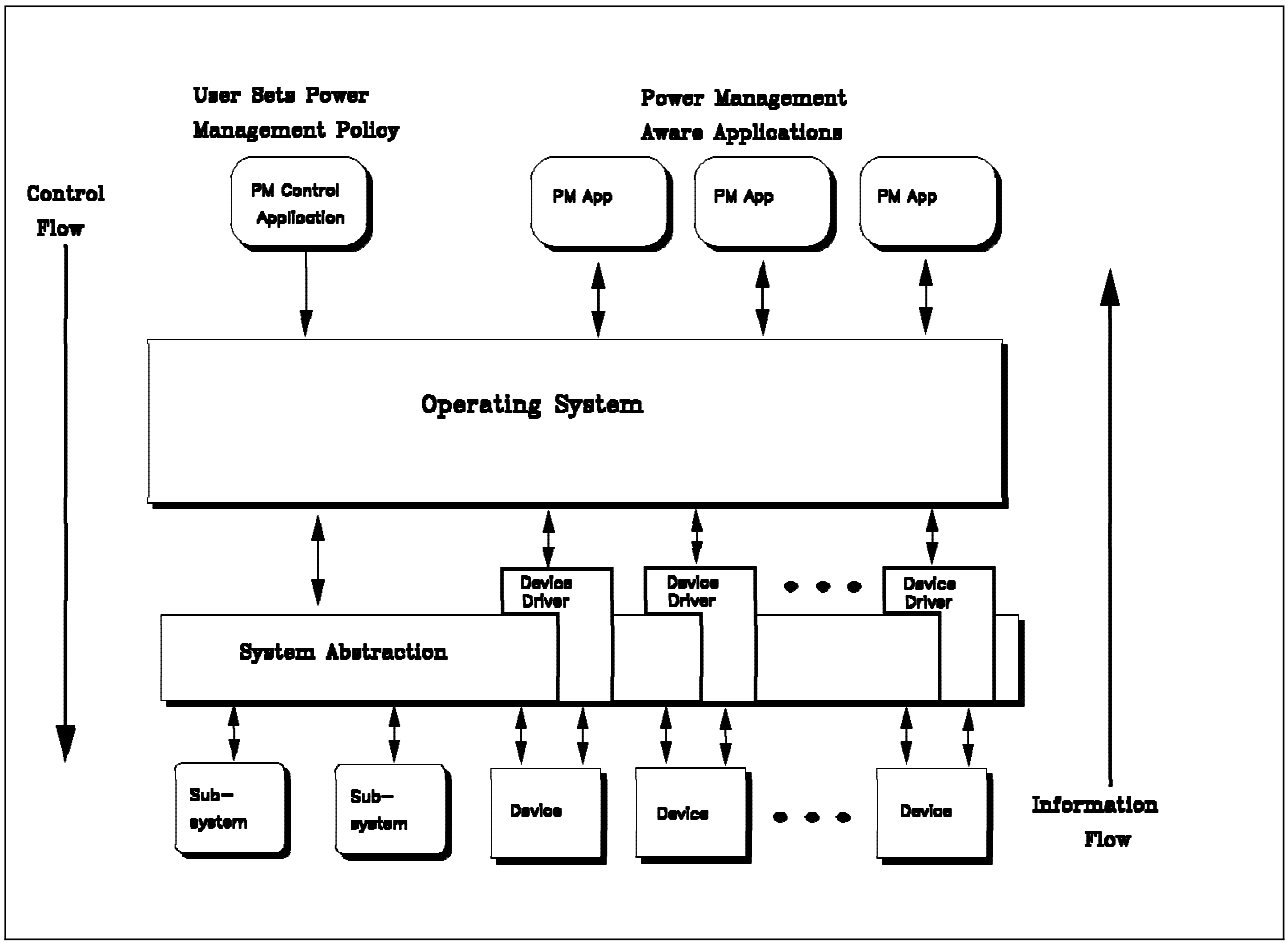

Figure 6. Macro Power Management Model

- It is recommended that power-managed systems provide a software-controllable main power switch which would allow automatic hibernation when the system is powered off or when the battery supply is critically low in portable systems.
- It is recommended that the hibernation flag be kept on the media that holds the saved system state. This approach may avoid confusion if that media was either removed from the system or was not operational after the system had been hibernated.

Miscellaneous

- To accommodate suspend and hibernation requirements, the NVRAM provides the following fields:
  - A 1-byte field which could be used to hold the suspend flag (called _PM_MODE, see Section 5.5.5, "Map of NVRAM Data Structure"). If this field is used for the suspend flag, then verifying the contents of memory is very important. This is due to the possibility of system power being lost during the suspend time frame. If the system is rebooted, the flag in NVRAM would remain set. Another alternative would be to use a special purpose register to hold this flag, which would default to the reset state if system power was lost and then restored.
  - A 32-bit resume pointer field (a member of the struct _RESTART_BLOCK called RestartAddress, see Section 5.5.5, "Map of NVRAM Data Structure"). This field is used to tell firmware the correct starting point to jump to when passing control to system software.
  - A 32-bit pointer (also a member of the _RESTART_BLOCK called SaveAreaAddr, see Section 5.5.5, "Map of NVRAM Data Structure"). Some firmware implementations may require a fixed amount of free contiguous memory to resume from suspend. The operating system would be required to free this space before setting the suspend flag. This OS would use this pointer to tell firmware the start address of this space.

3.11.2 Device and Subsystem Power Management

Effective system power management depends on subsystems and devices which provide power management support. This usually means support for multiple power states and hooks for system software to control those states. Some devices and subsystems may implement micro power management techniques, in addition to any macro power management capabilities they provide. A good example of this is the PowerPC 603 processor's management of the floating-point unit. It will turn the floating-point unit off and on based on the instruction stream it is executing.

Requirements, recommendations, and miscellaneous information follow:

Requirements

- If a device or subsystem is micro power managed, then these techniques must not affect the correct operation of programs.
- When entering the suspend state, the CPU core must be the last device to power off and the first device to power on when resuming. The CPU core is the hardware necessary to execute instructions, and includes the processor, clock, cache, and primary processor bus.

Recommendations

- The following device and subsystem modes are recommended as a minimal set for macro power management. The definitions are intended as a general guideline and will vary depending on the device or subsystem.
  - *On*:
    - — The device or subsystem is fully powered and able to perform at maximum performance.
  - *Power Managed*:
    - — The device or subsystem is working, but some or all features may be functioning at reduced performance levels.
    - — Power is maintained.
    - — State information is retained.
  - *Low Power*:
    - — The device or subsystem is not operational, but can resume to the power-managed state quickly.
    - — Some power is maintained.
    - — Device state information is retained.
  - *Off*:
    - — The device or subsystem is not operational.
    - — The device or subsystem is powered off.
    - — State information is not retained.

Miscellaneous

- The use of micro-power-managed devices requires careful consideration to avoid incorrect program operation. Two areas of particular concern are making sure system buffers are written to disk before powering off logic which contains them, and device or subsystem response time while running time-critical applications.

3.11.3 Power Management Hardware and Abstraction Software Features

Requirements, recommendations, and miscellaneous information follow:

Requirements

- The processor timekeeping (e.g. Real-Time Clock in 601 and time base in other PowerPC processors) must be accounted for in the hardware and software systems. This implies the following:

/ – If the processor timekeeping functions are going to be slowed down as in the power-managed state,
/ then this frequency change must be accounted for when calculating time of day or elapsed time to an
/ interrupt.
/ – If the processor timekeeping functions are going to be stopped as in the power-managed -- off state,
/ then time-based interrupts (e.g. decrementer) and the time of day which must persist across the
/ power off period must be maintained in other hardware within the system.
/ • Each device must provide for its state to be restored.
/ • If a device's state registers are implemented as write only, then the device driver must maintain a shadow
/ copy of these registers.

Recommendations

/ • It is recommended that only systems implementing Open Firmware use the technique of slowing the
/ processor clock to manage power consumption. Switching the clock between its normal frequency and
/ off is acceptable.
• It is recommended that device registers be read/write for use with save and restore functions.
• It is recommended that components of the system provide idle and/or usage indications to the power management controller software.
/ • It is recommended that the system preserve security features during power management. For instance, if
/ the system has a mechanism
/ to check for a user password when powering on from an "off" state, then this facility should also exist
/ when powering on from a "hibernate" state.
• It is recommended that systems using DRAM be designed to allow the refresh rate to be slowed down or switched to a self-refresh mode. In the self-refresh mode, the memory controller may be shut down.
• For devices with counters, it is recommended that the internal counter's clock be gated off and not used except when required for the counter.
• It is recommended that memory controller designs have the following attributes:
  – All registers should be read/write.
  – Support CAS before RAS refresh in order to allow low-power DRAM refresh. This mode may be used in both normal and suspend mode.
  – Use RTC clock at 32 KHz as the refresh clock in suspend mode.
  – Tri-state all address, data and control signals except the memory interface in suspend mode.
  – Stop clock to all logic except memory interface in suspend mode.
• If a PCI bus is present, it is recommended that the PCI clock, which is programmable, be turned off in suspend mode.

/ ***Miscellaneous***

/ • Open Firmware will provide methods to query devices and subsystems about their power management
/ capabilities, including the frequency at which the processor will run in its different power-managed states.
/ An architected means to communicate this information to the operating system prior to implementing
/ Open Firmware does not exist.

3.12 Bi-Endian Support

Requirements and miscellaneous information follow:

Requirements

• A system must support two modes of operation: Big-Endian and Little-Endian.
• The hardware system design must provide the capabilities for a Big-Endian operating system (e.g. AIX) and its applications to run efficiently or for a Little-Endian operating system (e.g. Windows NT** or WP/OS) and its applications to run efficiently.
/ • A system must provide hardware which presents the data to the processor in the proper format in both
/ Endian modes.

/ • A system must include hardware which presents the data and addresses to memory and I/O in the
/ proper format in both Endian modes.

Miscellaneous

- A system that runs operating systems and applications software and uses data that are stored either Big-Endian (e.g. big end first -- most significant byte first) or Little-Endian (e.g. little end first -- least significant byte first) is referred to as "Bi-Endian."
- This Bi-Endian requirement does not mean that simultaneous operation of Big-Endian and Little-Endian, called "Mixed-Endian," is required. For instance, a Little-Endian operating system running a Big-Endian application without the aid of emulation and translation software is a Mixed-Endian environment. Eventually this Mixed-Endian capability may become practical, but currently the synchronization at mode switching time degrades performance.
- / Because the current PowerPC processors assume that multibyte scalars are placed in storage in Big-Endian byte order, there must be some translation somewhere in the system when the processor references objects (programs, data, etc.) which are Little-Endian. Refer to Appendix B, "Bi-Endian Design
/ Guidance," for a description of Bi-Endian design alternatives. Designs which are applicable to the
/ current set of processors (e.g. PowerPC 601, 603, 604, and 620) would have the following attributes:
/ – The designs would provide hardware which reverses the location of each byte in each doubleword
/ when in Little-Endian mode.
/ – The designs would provide hardware which modifies (e.g. returns the address to its original value)
/ the address of a scalar generated by the processor when in Little-Endian mode.
/ – The designs would provide these hardware functions together either on the path between memory
/ and I/O to the processor or on the path between the I/O devices and memory and the processor.

3.12.1 Software Support for Bi-Endian Operation

Requirements, recommendations, and miscellaneous information follow:

Requirements

- Each Little-Endian operating system must perform an Endian mode switch during the boot process.
- / The Endian mode switching instructions must be provided by the abstraction software for each Little-
/ Endian operating system.
- Software support for the processor capabilities to perform byte reversal operations must be provided.

Recommendations

- / It is recommended that software support both the Bi-Endian Memory and I/O design (refer to
/ Figure 56) and the Bi-Endian I/O design (refer to Figure 57).
- / It is recommended that device drivers and other service code be written in an Endian-neutral manner.

Miscellaneous

- At power on, or after a system reset, the processor is in Big-Endian mode. The particular instructions to perform an Endian mode switch are processor and hardware system dependent.
- In some cases software may have to byte-reverse data before sending it to an I/O device. For instance, Big Endian device drivers sending multibyte control data to Little-Endian registers on an I/O device must byte-reverse this data in the processor.
- | Load and store with byte reversal of 2- and 4-byte scalars are contained in the PowerPC architecture. Invoking these instructions should be supported by language syntax and compiler support. If the com-
| pilers for all languages do not support these forms of loads and stores, then the runtime library routines should supply services which perform the byte reversal.
- / "Endian-neutral." is defined as software which is aware of the Endianess of the system and is designed
/ for portability between different Endian systems. For a discussion of Endian-neutral programming tech-
/ niques see the IBM Technical report by Jim Gillig, *How to Create Endian Neutral Software for Porta-*

bility, TR54.837. This information was also published in the October and November 1994 issues of *Dr. Dobb's Journal*.

3.12.2 Big-Endian and Little-Endian Storage

This section shows examples of the storage of data and instructions in Big-Endian and Little-Endian formats. Miscellaneous information follows:

Miscellaneous

- Figure 7 shows an example of a C language data structure, *s*. The value in each byte is shown in the comments portion of this structure. The C language mapping rules will introduce padding for alignment independent of Endian mode. Table 6 shows this structure mapping as it would appear in memory or on recording media in Big-Endian format with the most significant byte (MSB) first. Table 7 shows the structure mapping in Little-Endian format. These two figures represent data as it would look on I/O storage media. Notice that the one-byte character string data is stored in the specified order, but for all other sizes, the data is stored with the MSB at the first address (n) and the next most significant byte at the next address (n+1) for Big-Endian and with the least significant byte first and MSB last for the Little-Endian mode.

```
struc{
        int     a;      /* 0x11121314                      word */
        doubl   b;      /* 0x2122232425262728              doubleword */
        int     c;      /* 0x31323334                      word */
        char    d[7];   /* 'A', 'B', 'C', 'D', 'E', 'F', 'G'  array of bytes */
        short   e;      /* 0x5152                          halfword */
        int     f;      /* 0x61626364                      word */
} s;
```

Figure 7. Example of a C Structure Showing Values of the Elements

Table 6. Structure s in Big-Endian Order

Address (Hex)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	8	**9**	**A**	**B**	**C**	**D**	**E**	**F**
A000	11	12	13	14				
A008	21	22	23	24	25	26	27	28
A010	31	32	33	34	'A'	'B'	'C'	'D'
A018	'E'	'F'	'G'		51	52		
A020	61	62	63	64				

Table 7. Structure s in Little-Endian Order								
Address (Hex)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	8	**9**	**A**	**B**	**C**	**D**	**E**	**F**
A000	14	13	12	11				
A008	28	27	26	25	24	23	22	21
A010	34	33	32	31	'A'	'B'	'C'	'D'
A018	'E'	'F'	'G'		52	51		
A020	64	63	62	61				

- Figure 8 shows a PowerPC assembly language code fragment which references the structure *s*. Table 8 shows this instruction stream with addresses resolved in Big-Endian format on recording media or memory. Notice that the fields of the instructions are placed within the bytes of a word for readability and not to signify where the actual data is placed. Table 9 shows this instruction stream with addresses resolved in Little-Endian memory or recording media. Notice that the instruction stream has been written with the fields ordered backwards within each word to signify the Little-Endian storage of these instructions. Notice that in both instruction streams the resolved reference addresses for quantities in structure *s* are identical.

```
        li      r5,6
        li      r6,8
loop:
        cmplwi  r5,0
        beq     done
        lbz     r4,d(r5)
        slw     r7,r6,r7
        or      r7,r7,r4
        subi    r5,2
        b       loop
done:
        stw     r7,f
```

Figure 8. Example of an Assembly Language Code Fragment

Table 8. Instructions in Big-Endian Order								
Address (Hex)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	8	**9**	**A**	**B**	**C**	**D**	**E**	**F**
B000	li	r5,	6		li	r6,	8	
B008	loop:	cmplwi	r5,	0	beq	B024		
B010	lbz	r4,	A014	(r5)	slw	r7,	r6,	r7
B018	or	r7,	r7,	r4	subi	r5,	2	
B020	b	B008			done:	stw	r7,	A020

Table 9. Instructions in Little-Endian Order								
Address (Hex)	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
	8	**9**	**A**	**B**	**C**	**D**	**E**	**F**
B000		6	r5,	li		8	r6,	li
B008	0	r5,	cmplwi	loop:			B024	beq
B010	(r5)	A014	r4,	lbz	r7	r6,	r7,	slw
B018	r4	r7,	r7,	or		2	r5,	subi
B020			B008	b	A020	r7,	stw	done:

3.13 Multiprocessor Considerations

/ The *PowerPC Reference Platform Specification* is intended to also support the implementation of symmetric
/ multiprocessor (SMP) systems. The main differences between uniprocessor and SMP systems are inter-
/ processor communications, inter-processor synchronization, I/O interrupt redirection and L2 cache recom-
/ mendations. Storage access ordering and memory coherence are also important elements of a functional
/ SMP system. Section A.6, “Symmetric Multiprocessor,” provides a detailed implementation example of an
/ SMP system.

/ Requirements and recommendations follow:

/ ***Requirements***

/ • Inter-processor synchronization: The *lwarx, stwcx, eieio* and *sync* instructions must be used to ensure
/ synchronization among processors, when required. See the *The PowerPC Architecture* for details.
/ • TLB synchronization: The TLB Invalidate Entry and TLB Synchronize instructions must be used to
/ synchronize TLBs among processors. The TLB Invalidate All instruction is optional.
/ • Time Base or Real-Time Clock: A means to synchronize the Time Base or Real-Time Clock among
/ processors must be provided.
/ • Reconfiguration: The system must provide, as part of the boot process, a means to vary a processor
/ (and associated dedicated second level cache, if any) into and out of the configuration.
/ • A processor not in the configuration must not be allowed to affect the operation of processors which are
/ in the configuration.
/ • All processors in the configuration must have equal access to system memory.
/ • All processors in the configuration must have equal access to all I/O devices and adaptors.
/ • All processors in the configuration must be of the same type (e.g. 604) and speed (e.g. 100 MHz).
/ • All L2 caches in the configuration must be of the same type (e.g. in-line, copy-back) and size (e.g. 1
/ MB).
/ • If an in-line L2 cache is used, it must support one reservation as defined for the *lwarx* and *stwcx*
/ instructions.

/ ***Recommendations***

/ • A dedicated, in-line, copy-back L2 cache per processor is strongly recommended.
/ • It is recommended that the I/O subsystem be designed to scale with the number of processors. If the
/ processors are upgradable, it is recommended that the I/O subsystem have the capability to scale with
/ relative processor performance as well.
/ • It is recommended that multiprocessing systems contain a minimum of 4 KB of Non-volatile Memory
/ per processor.

3.13.1 MP Interrupts

Figure 52 in Section A.6, “Symmetric Multiprocessor,” shows one possible implementation of MP interrupts.

Requirements, recommendations, and miscellaneous information follow:

Requirements

- The system must support a total of at least 16 inter-processor and I/O interrupt requests.
- There must be a means to redirect any I/O interrupt request to a specific processor or to all processors.
- There must be a means to allow software to generate an individual interrupt request from any processor to any processor (inter-processor interrupts).
- There must be a means to identify the interrupt source.
- The process of handling an interrupt on one processor must not be allowed to disrupt other processors.

Recommendations

- It is recommended that the hardware abstraction software isolate the operating system from the MP interrupt hardware. One possible abstraction interface is described below.
- One inter-processor interrupt request per processor plus one I/O interrupt request per native I/O device plus at least one I/O interrupt request per add-in card slot are recommended.
- A means to allow software to generate a floating system-wide interrupt request is recommended.
- A means to assign a priority level to each interrupt request is recommended. (Required for AIX.)
- A means to assign a priority level to each processor is recommended. (Required for AIX.)
- A means to allow software to poll for pending interrupts is recommended.
- To reduce the frequency of interrupts and to avoid unnecessarily restrictive requirements for low-latency interrupt handling, it is recommended that I/O devices, particularly high-speed serial devices, be buffered.
- Interrupt sharing is not recommended (except when the devices sharing interrupts also share the same device driver and interrupt priority).
- Inter-processor interrupts should be low-latency.

Miscellaneous

- Intel’s 82489DX Advanced Programmable Interrupt Controller meets all the requirements and the first six recommendations listed above. One 82489DX integrated circuit is used per processor. (Note: while PowerPC based systems can use the 82489DX integrated circuit, the local unit and I/O unit logic cores which make up the 82489DX are not licensed for use within PowerPC processors or within chip sets designed for use with PowerPC processors.)
- A possible interrupt abstraction interface:
  - GetInterruptVector
  - EnableInterrupt
  - DisableInterrupt
  - GenerateInterrupt: Software-generated interrupt.
  - RaiseIRQL: Raises the processor’s priority level.
  - LowerIRQL: Lowers the processor’s priority level.
- Future directions for MP interrupts:
  - Interrupts should be associated with each request, not with a card slot, device adaptor or device. Advantages:
    - Allows software to assign a unique interrupt vector to all requests to a particular device. Each device (e.g. fixed disk, CD-ROM, tape) connected to a SCSI controller could have a unique interrupt vector.
    - Allows software to assign different interrupt priority levels to requests to the same device or to the same device adaptor. Requests associated with high-priority tasks could have higher priority interrupt levels than requests associated with lower-priority tasks.

/ – Elimination of sideband signals (e.g. INTA, INTB) for interrupts. Adding an interrupt enqueue
/ command to the PCI bus, for example, could allow a device (or device adaptor) to set an interrupt
/ pending bit within the interrupt controller without the use of sideband signals. Using the same inter-
/ face to enqueue the interrupt as to transfer the data also eliminates some data coherency concerns by
/ ensuring that data buffers are flushed before the interrupt is enqueued.
/ – Adding interrupt enqueue and interrupt dequeue commands to the PowerPC processor bus (as
/ address-only requests) could allow the interrupt controller to be embedded within each processor
/ chip without requiring additional I/O or incurring the additional latency of a serial interrupt inter-
/ face.

3.14 Alignment Considerations

This section defines the requirements and recommendations for use and support of aligned and unaligned operations. It provides directions which will allow PowerPC Reference Platform systems to run applications and operating systems which have a Power legacy and points out changes in that legacy which will have to be accounted for in these PowerPC based machines. Requirements, recommendations, and miscellaneous information follow:

Requirements

/ • Noncachable operations must be handled by other system components in a manner defined in Table 10.

Recommendations

• Because some unaligned operations may have performance penalties, it is strongly recommended that all PowerPC software be restricted to using size-aligned load/store operations.
/ • It is recommended that noncachable load and store multiple and move assist instructions need not be
/ supported by the components of the system outside of the processor.
/ • It is recommended that floating-point load or store instructions which are not word aligned need not be
/ supported by the components of the system outside of the processor.

Table 10 (Page 1 of 2). Specification for Handling of Alignment

Instructions as Defined in the PowerPC Architecture	**Aligned**		**Unaligned**	
	BE	**LE**	**BE**	**LE**
Fixed-Point Load Instructions	P	P	P	E
Fixed-Point Store Instructions	P	P	P	E
Load and Store with Byte Reversal Instructions	P	P	P	E
Fixed-Point Load and Store Multiple Instructions	D	E	E*	E
Fixed-Point Move Assist Instructions (e.g. String)	D	E	D	E
Storage Synchronization Instructions *lwarx* and *stwcx*	P	P	B	B
Floating-Point Load Instructions	P	P	W*	E

Table 10 (Page 2 of 2). Specification for Handling of Alignment				
Instructions as Defined in the PowerPC Architecture	**Aligned**		**Unaligned**	
	BE	**LE**	**BE**	**LE**
Floating-Point Store Instructions including the optional instruction *stfiwx*	P	P	W*	E

Note:

- For more information refer to the PowerPC User Instruction Set in *The PowerPC Architecture*, Book I.
- Any unaligned operation which crosses a protection boundary may cause the alignment interrupt to be invoked. Refer to Section 5.5.6, PowerPC Operating Environment, in *The PowerPC Architecture*, Book III.
- Big-Endian (BE); Little-Endian (LE)

B Boundedly undefined or a system alignment error handler invoked by the processor
D Discouraged, but cachable operations are currently supported by the PowerPC processors
E System alignment or system error handler invoked by the processor
P Processed normally
W Word alignment of doublewords is only unaligned case allowed
***** The 601 Processor processes unaligned forms of these instructions, but other processors do not

Miscellaneous

- Size alignment will ensure that applications will run with the highest possible performance on all PowerPC hardware implementations. The degradation of performance is dependent upon the particular processor used and the implementation approach of other hardware system components. Size alignment is defined as follows:

| **Halfword** The address is divisible by 2 and is in the form B'x...xxx0'

| **Word** The address is divisible by 4 and is in the form B'x...xx00'

| **Doubleword** The address is divisible by 8 and is in the form B'x...x000'

/ • Table 10 defines the supported and unsupported forms of the instructions for cachable and noncachable / operations. Note that some processors may have implemented a superset of this support (refer to Section 3.17, "PowerPC Architecture Features Not Recommended"). Cachable operations are handled within the PowerPC processors. Noncachable operations are either 1) cache-inhibited loads and stores to or from memory or 2) loads and stores to or from I/O devices. Notice that some operations are "discouraged." PowerPC Reference Platform-compliant systems may support these operations as a transition from earlier architectures. Future versions of the PowerPC architecture and processors may not support these instructions. Software which depends upon these instructions may not run on future machines or may suffer a performance impact if these instructions are used.

/ 3.15 Support for Loads and Stores to System I/O Bus

/ This section defines the hardware system support for loads and stores to the system I/O bus. Requirements, recommendations, and miscellaneous information follow:

Requirements

/ • If the processor has an 8-byte data bus, then the system must support all processor-generated loads and / stores to I/O buses for 1-, 2- and 4-byte transfers which do not span a word boundary and 3-byte trans- / fers which do not span a word boundary and are the result of the processor breaking a larger load or / store at the doubleword boundary.

/ • If the processor has a 4-byte data bus, then the system must support all processor-generated loads and
/ stores to I/O buses for 1-, 2- and 4-byte transfers which do not span a word boundary and 3-byte trans-
/ fers which do not span a word boundary and are the result of the processor breaking a larger load or
/ store at the word boundary.
• If a vendor implements a 2-byte bus, then the hardware must accommodate the 4-byte I/O transfers generated on a system. Four-byte transfers would have to be broken into two 2-byte transfers.

Recommendations

• It is recommended that a system not support references which span a word boundary for loads and stores to system I/O buses which are only one word (4 bytes) wide.
/ • It is recommended that a system support 8 byte transfers, if these transfers will result in more efficient
/ utilization of the I/O bus.

Miscellaneous

/ • Table 11 shows the addresses which the processor generates and the required and recommended system
/ responses for I/O to a 4-byte bus. The following information describes this table:
– Combinations of transfer sizes and starting addresses which are presented by the processor to the bus
/ bridge and are required transfers for a word-wide bus are marked as “Support.”
/ – Those transfers which are not recommended for a one-word-wide bus (e.g. transfers which span a word boundary) are marked as “Undefined results.”
– Some combinations of size and starting address will not be generated by the PowerPC processor and are marked as “N/A.” For instance, the PowerPC processors break any request at a doubleword boundary. A word request starting at B'x...x111' becomes a 1-byte transfer from B'x...x111' and a
/ 3-byte transfer from B'x..x1000'. The column of this table labeled “4-byte data bus” represents the
/ actions of a 603 processor. For the 603 and for any future processor with a word-wide data bus, unaligned transfers which cross a word boundary are also broken into two pieces.
– Note that 3-byte transfers occur only as a result of these word and doubleword split transfers or as a result of move assist (e.g. string) instructions. String instructions are not recommended in a PowerPC Reference Platform system. Three-byte transfers due only to string operations which would not cross the word boundary are marked as “undefined results (string).”
– Five-, 6-, and 7-byte transfers are generated only as a result of floating-point doubleword unaligned loads and stores which are implemented only in the 601 processor and are not recommended for PowerPC Reference Platform systems. These transfers are marked as “601 undefined results.”
/ – For any “undefined results,” a system design may provide an error response, may provide undefined data, or may implement support for some or all of these cases. A more complex bus bridge could be designed to buffer requests that spanned a word and send them onto the I/O bus as two requests.
• An 8-byte I/O bus would support the required loads and stores defined in this table. In addition, some “undefined results” conditions in this table such as unaligned words and 8-byte transfers could be handled by the 8-byte bus.
• The terminology used in Table 11 is summarized below:

Term	Meaning
Support	A bus bridge must support this transfer address and size
Undefined Results	A bus bridge may support, may give an error response, or may give undefined data
N/A	A PowerPC processor will not generate this combination of address and size to the PowerPC processor bus
Undefined Results (String)	This combination of address and length will only be produced by a string operation and the bus bridge may support, may give an error message, or may give undefined data

601 Undefined Results This combination of address and length will only be generated from a 601 processor and the bus bridge may support, may give an error response, or may give undefined data

Table 11 (Page 1 of 2). Processor-Generated Load and Store Addresses to System I/O Buses

Transfer Size	CPU Address	Response For System With Processor Which Has	
		8-Byte Data Bus	4-Byte Data Bus
Byte	B'x...x000'	Support	Support
	B'x...x001'	Support	Support
	B'x...x010'	Support	Support
	B'x...x011'	Support	Support
	B'x...x100'	Support	Support
	B'x...x101'	Support	Support
	B'x...x110'	Support	Support
	B'x...x111'	Support	Support
Halfword	B'x...x000'	Support	Support
	B'x...x001'	Support	Support
	B'x...x010'	Support	Support
	B'x...x011'	Undefined Results	N/A
	B'x...x100'	Support	Support
	B'x...x101'	Support	Support
	B'x...x110'	Support	Support
	B'x...x111'	N/A	N/A
3-byte	B'x...x000'	Support	Support
	B'x...x001'	Undefined Results (String)	Support
	B'x...x010'	Undefined Results	N/A
	B'x...x011'	Undefined Results	N/A
	B'x...x100'	Undefined Results (String)	Support
	B'x...x101'	Support	Support
	B'x...x110'	N/A	N/A
	B'x...x111'	N/A	N/A

Table 11 (Page 2 of 2). Processor-Generated Load and Store Addresses to System I/O Buses			
Transfer Size	**CPU Address**	**Response For System With Processor Which Has**	
		8-Byte Data Bus	**4-Byte Data Bus**
Word	B'x...x000'	Support	Support
	B'x...x001'	Undefined Results	Undefined Results
	B'x...x010'	Undefined Results	Undefined Results
	B'x...x011'	Undefined Results	Undefined Results
	B'x...x100'	Support	Support
	B'x...x101'	N/A	N/A
	B'x...x110'	N/A	N/A
	B'x...x111'	N/A	N/A
5-byte	Any	N/A (601 Undefined Results)	N/A
6-byte	Any	N/A (601 Undefined Results)	N/A
7-byte	Any	N/A (601 Undefined Results)	N/A
/ Doublewords	B'x...x000'	Undefined Results*	Undefined Results*
	B'x...x001'	N/A	N/A
	B'x...x010'	N/A	N/A
	B'x...x011'	N/A	N/A
	B'x...x100'	N/A	N/A
	B'x...x101'	N/A	N/A
	B'x...x110'	N/A	N/A
	B'x...x111'	N/A	N/A
/ **Note:** * Recommend support, if more efficient utilization of the bus will result.			

3.16 Cache-Inhibited Loads and Stores to System Memory

/ This section describes the hardware system support for cache-inhibited loads and stores to System Memory. Requirements, recommendations and miscellaneous information follow:

Requirements

/ • If the processor has an 8-byte data bus, then the system must support all processor-generated loads and
/ stores to system memory for 1-, 2-, 4- and 8-byte transfers which do not span a doubleword boundary
/ and 3-byte transfers which do not span a doubleword boundary and are the result of the processor
/ breaking a larger load or store at the doubleword boundary.
/ • If the processor has a 4-byte data bus, then the system must support all processor-generated loads and
/ stores to system memory for 1-, 2- and 4-byte transfers which do not span a word boundary; 3-byte
/ transfers which do not span a word boundary and are the result of the processor breaking a larger load
/ or store at the word boundary; and if the processor is in 8-byte mode, 8-byte transfers which do not
/ span a doubleword.

Recommendations

- It is recommended that systems not support references which span a doubleword boundary for cache-inhibited loads and stores to system memory.

Miscellaneous

- Table 12 shows the addresses which the processors generate and the required response of the system for loads or stores to system memory. The following describes this table:
  - Combinations of transfer sizes and starting addresses which are presented by the processor to the memory controller and are required transfers are marked as "Support."
  - Some combinations of transfer size and starting address will not be generated by the PowerPC processor and are marked as "N/A.." For instance, the PowerPC processors break any request at a doubleword boundary. A word request starting at B'x...x111' becomes a 1-byte transfer from B'x...x111' and a 3-byte transfer from B'x..x1000'. The column in this table marked "4-byte data bus" represents the actions of the 603 processor. For the 603 and for any future processor with a word-wide data bus, unaligned transfers which cross a word boundary are also broken into two pieces.
  - Note that 3-byte transfers occur only as a result of these word and doubleword split transfers or as a result of move assist (e.g. string) instructions which are not recommended in a system. Three-byte transfers which are generated only as a result of string operations are marked as "Undefined results (string)."
  - Five-, 6-, and 7-byte transfers are generated only as a result of doubleword unaligned loads and stores which are implemented only in the 601 processor. Unaligned Floating-Point loads and stores are not recommended for systems and are marked as "601 undefined results."
  - For any "undefined results," an implementation may provide an error indication, may provide undefined data, or may support these transfers.
- The terminology used in Table 12 is summarized below:

Term	Meaning
Support	A memory controller must support this transfer address and size
N/A	A PowerPC processor will not generate this combination of address and size to the PowerPC processor bus
Undefined Results (String)	This combination of address and length will only be produced by a string operation and the memory controller may support, may give an error message, or may give undefined data
601 Undefined Results	This combination of address and length will only be generated from a 601 processor and the memory controller may support, may give an error response, or may give undefined data

Table 12 (Page 1 of 2). Cache-Inhibited Load and Store Addresses to System Memory

Transfer Size	CPU Address	Response For System With Processor Which Has	
		8-Byte Data Bus	**4-Byte Data Bus**
Byte	B'x...x000'	Support	Support
	B'x...x001'	Support	Support
	B'x...x010'	Support	Support
	B'x...x011'	Support	Support
	B'x...x100'	Support	Support
	B'x...x101'	Support	Support
	B'x...x110'	Support	Support
	B'x...x111'	Support	Support
Halfword	B'x...x000'	Support	Support
	B'x...x001'	Support	Support
	B'x...x010'	Support	Support
	B'x...x011'	Support	N/A
	B'x...x100'	Support	Support
	B'x...x101'	Support	Support
	B'x...x110'	Support	Support
	B'x...x111'	N/A	N/A
3-byte	B'x...x000'	Support	Support
	B'x...x001'	Undefined results (string)	Support
	B'x...x010'	Undefined results (string)	N/A
	B'x...x011'	Undefined results (string)	N/A
	B'x...x100'	Undefined results (string)	Support
	B'x...x101'	Support	Support
	B'x...x110'	N/A	N/A
	B'x...x111'	N/A	N/A
Word	B'x...x000'	Support	Support
	B'x...x001'	Support	N/A
	B'x...x010'	Support	N/A
	B'x...x011'	Support	N/A
	B'x...x100'	Support	Support
	B'x...x101'	N/A	N/A
	B'x...x110'	N/A	N/A
	B'x...x111'	N/A	N/A
5-byte	Any	N/A (601 Undefined results)	N/A
6-byte	Any	N/A (601 Undefined results)	N/A

Table 12 (Page 2 of 2). Cache-Inhibited Load and Store Addresses to System Memory			
Transfer Size	**CPU Address**	**Response For System With Processor Which Has**	
		8-Byte Data Bus	**4-Byte Data Bus**
7-byte	Any	N/A (601 Undefined results)	N/A
/ Doublewords	B'x...x000'	Support	Support*
	B'x...x001'	N/A	N/A
	B'x...x010'	N/A	N/A
	B'x...x011'	N/A	N/A
	B'x...x100'	N/A	N/A
	B'x...x101'	N/A	N/A
	B'x...x110'	N/A	N/A
	B'x...x111'	N/A	N/A
Note: * N/A when the 603 is running in 4-byte data bus mode			

3.17 PowerPC Architecture Features Not Recommended

Some PowerPC Architecture features need not be implemented in a PowerPC Reference Platform-compliant system. Some of these features are transition or carryover from an earlier Power architecture. Other of these features are not supported consistently in PowerPC implementations. The intent of recommending that PowerPC Reference Platform-compliant systems not use these features is to position these systems for support across the full family of emerging processors. The features which are not recommended for implementation in a PowerPC Reference Platform-compliant system are described below.

3.17.1 Unaligned Little-Endian Scalar Operations

Recommendations and miscellaneous information follow:

Recommendations

- Because of performance impacts, software running on PowerPC Reference Platform systems is strongly discouraged from using unaligned Little-Endian scalar operations.

Miscellaneous

- The current PowerPC processors do not support any unaligned Little-Endian load or store operations. The PowerPC processor will interrupt to the system alignment handler for resolution.

3.17.2 Unaligned Little-Endian Multiple Scalar Operations

Recommendations and miscellaneous information follow:

Recommendations

- Because of performance impacts, software running on PowerPC Reference Platform systems is strongly discouraged from using unaligned Little-Endian multiple scalar operations.

Miscellaneous

- The current PowerPC processors do not support any Little-Endian multiple scalar load or store operations. This includes load and store multiple and load and store string operations. The PowerPC processor will interrupt to the system alignment handler for resolution.

3.17.3 Direct-Store Segments

Requirements and miscellaneous information follow:

Requirements

- As a minimum, operating system software and PowerPC Reference Platform systems must support ordinary storage segments.
- PowerPC Reference Platform-compliant operating systems must not depend on direct-store segments
/ when running on PowerPC Reference Platform machines.

Miscellaneous

- Section 12.6 of the *The PowerPC Architecture* defines the architecture of direct-store segments as "a mapping of effective addresses onto an external address space, typically an I/O bus." This capability allows for synchronous I/O to other address spaces. Direct-store segments need not be implemented in PowerPC Reference Platforms. As an option in unique environments tied to specific expansion bus support, some operating systems and some systems may support both ordinary storage segments and direct-store segments.

3.17.4 Load and Store String

Requirements, recommendations, and miscellaneous information follow:

Requirements

- Software for PowerPC Reference Platform implementations must not use the load and store string instructions for cache-inhibited access to either I/O or System Memory.

Recommendations

/ - It is recommended that a memory controller/bus bridge implementation for a system reduce complexity and cost by not supporting the 3-byte transfers generated only by load and store string operations.

Miscellaneous

- Section 3.3 of the *The PowerPC Architecture* defines four instructions which load a string of 1 through n bytes into consecutive registers 4 bytes at a time. In future PowerPC processor implementations these instructions are likely to have greater latency and take longer to execute, perhaps much longer, than a sequence of individual load or store instructions. Processors currently fully support these instructions. These instructions are not efficient for noncachable operations and are the only instructions which generate 3-byte loads and stores. The specific transfers for the processor to I/O bus and for cache-inhibited accesses of the processor to System Memory are defined in Table 11 and Table 12.

3.17.5 Load and Store Multiple

Requirements, recommendations, and miscellaneous information follow:

Requirements

- Software for PowerPC Reference Platform implementations must not use the load and store multiple instructions for cache-inhibited access to I/O or System Memory.

Recommendations

/ - It is recommended that a memory controller/bus bridge implementation for a system treat cache-inhibited load and store multiple instructions the same as word or doubleword transfers because they appear as a series of word, or in the case of storage gathering, doubleword transfers.

Miscellaneous

- Section 3.3 of the *The PowerPC Architecture* defines the load and store multiple instructions which will
/ move one to 32 words to or from consecutive registers. The architecture states that if the words are not aligned, the system alignment error handler should be invoked or the results might be undefined. As a transition feature, the MPC601 processor allows these instructions to be unaligned and treats them as a series of unaligned word fetches or stores. In future PowerPC processor implementations these instructions are likely to have greater latency and take longer to execute, perhaps much longer, than a sequence of individual load or store instructions. Processors currently fully support these instructions. Some processors implement "storage gathering" which collects adjacent aligned quantities into single larger bus transfers. In this way, a store multiple to an I/O bus could become a series of doubleword transfers which may not be supported on the bus. The specific transfers for the processor to I/O bus and for cache-inhibited accesses of the processor to System Memory are defined in Table 11 and Table 12.

3.17.6 Unaligned Floating-Point Load and Store

Requirements, recommendations, and miscellaneous information follow:

Requirements

/ - Software for PowerPC Reference Platform implementations must not use the unaligned load and store
/ floating-point doubleword instructions for cache-inhibited access to I/O or System Memory.

Recommendations

- It is strongly recommended that software not use unaligned load and store floating-point operations.
/ - It is recommended that a memory controller/bus bridge implementation for a system reduce complexity
/ and cost by not supporting the 5-, 6- and 7-byte transfers generated only by unaligned load and store
/ floating-point doubleword instructions.

Miscellaneous

/ - The architecture requires word and doubleword floating-point load and store instructions to be word aligned as defined in *The PowerPC Architecture*, Section 13.5.6. As a transition feature, the 601 processor allows these instructions to be unaligned. An unaligned form of a doubleword floating-point noncachable load or store would generate 5-, 6- or 7-byte transfers as the doubleword is broken at the doubleword boundary.