PA-RISC 2.0 Firmware Architecture Reference Specification

Version 0.36

Printed in U.S.A. March 7, 1999

Notice

4. Operating System Interfaces

The I/O Architecture defines several interfaces between an operating system and other entities. Some interfaces can be described as entry points into the operating system. The I/O Architecture specifies the circumstances under which the entry point is invoked and the system state that must exist at the time of the invocation. The entry points may be triggered by hardware events or called by PDC or ISL.

Nine entry points into operating system software are defined by the I/O Architecture, as follows:

- **Boot (OS_BOOT)**. This is the code loaded and invoked by ISL when the operating system is booted. Its purpose is to perform the initial configuration of the operating system.
- **Rendezvous (OS_RENDEZ)**. This code is invoked when a rendezvous interrupt on EIR{0} is received.
- **TOC Handler (OS_TOC)**. This code is invoked by PDCE_TOC if conditions necessary to invoke OS_TOC are satisfied. Its purpose is to analyze system errors and effect recovery (if possible).
- **HPMC Handler (OS_HPMC)**. This code is invoked by PDCE_CHECK if conditions necessary to invoke OS_HPMC are satisfied. Its purpose is to complete fault analysis and effect recovery (if possible).
- **LPMC Handler (OS_LPMC)**. This code is invoked by PDCE_CHECK to report error conditions completely corrected by PDCE_CHECK. Its purpose is only to log error information provided in PIM and return to the interrupted process.
- **First-Level Interrupt Handler (OS_FLIH)**. OS_FLIH is triggered by the Interval Timer or when the processor receives an external interrupt message. It is the interruption handling routine for external interrupts.
- **Local Powerfail Warning (OS_PFW_LOCAL)**. OS_PFW_LOCAL is triggered when BUS_POW_WARN on the central bus is asserted. It is the interruption handling routine for a power failure interrupt.
- **Local Powerfail Recovery (OS_PFR)**. OS_PFR is invoked by PDCE_RESET if powerfail recovery is possible. Its purpose is to reconstruct the system state to that which prevailed at the time the power failure interrupt occurred.
- **Remote Powerfail Warning (OS_PFW_REMOTE)**. OS_PFW_REMOTE is triggered by an external interrupt to EIR{1}, caused by the assertion of BUS_POW_WARN on a remote bus. This code may be included within OS_FLIH, and its purpose is to handle the remote powerfail warning and to effect recovery when power returns to the remote bus.

The internal organization of an operating system is outside the scope of the I/O Architecture. However, it is essential that the environment that needs be established at each operating system interface, and a set of actions/rules that each entry point should perform/follow be architected, to guarantee proper system functionality. These requirements are presented under the descriptions for the various entry points.

Sections 4.1 through 4.6 cover OS_BOOT, OS_RENDEZ, OS_TOC, OS_HPMC, OS_LPMC, and OS_FLIH, in that order. Section 4.7 gives an overview of the recovery actions in different power failure situations and the three operating system entry points which are related to powerfail. The three major powerfail and recovery entry points are then presented: OS_PFW_LOCAL in Section 4.8, OS_PFR in Section 4.9, and OS_PFW_REMOTE in Section 4.10.

4.1 Boot (OS_BOOT)

OS_BOOT code is loaded and invoked by ISL when the operating system is booted. Its purpose is to perform the initial configuration of the operating system.

4.1.1 Interface to OS_BOOT

A description of the conditions that need to be met before ISL invokes OS_BOOT is given under the OS_BOOT interface description in Section 2.2, PDC Entry Points.

When invoked, OS_BOOT must be aware that there may be modules on the central bus as well as on remote busses that need to be reset, in order for OS_BOOT to function correctly.

4.1.2 The Role of OS_BOOT

OS_BOOT must be aware of the following rules in order to successfully perform initial system configuration.

- OS_BOOT is responsible for completing the allocation of system address space. With the single exception of the Initial Memory Module (IMM), whose SPA space must remain at 0x00000000, the operating system is free to change the system address space allocations made by PDC.
- OS_BOOT must never reset the IMM, nor must it disable the SPA of the IMM.

  If OS_BOOT finds a processor-dependent memory module which is a satellite of the IMM, that satellite must not be reset, must not have 0 written to its IO_SPA register, nor may PDC_IODC (ARG1=2) be called with it as the target.

  If OS_BOOT finds a processor-dependent memory module which is a satellite of a module other than the IMM, that module must be reset or must have 0 written to its IO_SPA register, before its SPA base can be changed by calling PDC_IODC (ARG1=2) for the primary memory module of the interleave group.

  OS_BOOT must never call PDC_IODC (ARG1=2) for a processor-dependent memory module which has a maximum SPA space size of zero, as the result may be the creation of SPA conflicts and/or holes in memory SPA.
- If OS_BOOT decides to salvage memory module(s) not configured by PDC, then it must be aware that processor-dependent memory modules and architected memory modules which failed ENTRY_TEST cannot be salvaged.
- OS_BOOT must configure the HPAs of modules on the central bus, to the central bus physical address space. This address space is allocated for the central bus, and cannot be assigned to other busses.

PROGRAMMING NOTE

Modules on the central bus may be factory configured to a fixed HPA, or they may have a relocatable HPA. In any case, the HPAs of modules on the central bus are never relocated, and always remain in the central bus physical address space.

- OS_BOOT must assign HPA space to all busses in the system that are powered. If there are any modules in the system that require SPA space, then it must satisfy their requirements.

  If OS_BOOT encounters a critical bus that does not have power, it must wait until the bus has power before it proceeds.

PROGRAMMING NOTE

OS_BOOT may notify the operator that it is waiting by writing a message to the console or to the front panel display.

If a non-critical bus without power is encountered, OS_BOOT should notify the operator that this non-critical bus is down. It should ask the operator if the boot can proceed without the bus. It will not be possible to power the bus up later and expect the operating system to do configuration at that point. The operator has two choices:

- Power the bus up when the console message is received.
- Notify the system that this bus will never come up; OS_BOOT can proceed at this point.

If no console is available, an operating system specific timeout can be used in lieu of an operator response.

- OS_BOOT must call PDC_COPROC to determine which coprocessors are present and functional. Since the state of all coprocessors is HVERSION dependent after calling PDC_COPROC, a state restore sequence of a valid coprocessor state must be used after the PDC_COPROC call to enable use of the coprocessor by the OS.
- OS_BOOT must call PDC_PIM (ARG1=3) at least once.

4.2 Rendezvous (OS_RENDEZ)

A processor will rendezvous (enter OS_RENDEZ code) if it is not selected as a monarch when competing with other processors in a multiprocessor configuration. Processors which are not selected as the monarch wait for a rendezvous interrupt on EIR{0}. On receipt of a rendezvous interrupt, a processor checks that the MEM_RENDEZ word in Page Zero is nonzero and the first word of the code at the location pointed to by MEM_RENDEZ is nonzero, and executes the OS_RENDEZ code.

4.2.1 Architected Interface to OS_RENDEZ

A description of the conditions that need to be met before PDCE_RESET invokes the OS_RENDEZ code is given under the OS_RENDEZ interface description in Section 2.2, PDC Entry Points.

4.3 TOC Handler (OS_TOC)

OS_TOC code is established when the operating system is first initialized. It contains a generalized error recovery routine. If an operator determines that the system has a problem, a TOC can be initiated from the control console or TOC button. This triggers PDCE_TOC, which then invokes OS_TOC. The OS_TOC code performs whatever recovery is possible. The operating system is not required to recover from TOCs.

4.3.1 Interface to OS_TOC

A description of the conditions that need to be met before PDCE_TOC invokes OS_TOC is given under the OS_TOC interface description in Section 2.2, PDC Entry Points.

4.3.2 The Role of OS_TOC

- OS_TOC must call PDC_PIM (ARG1=4) at least once.
- If it is determined that OS_TOC can return to the interrupted process, then OS_TOC must check certain validity fields in PIM to ensure that the CPU state in PIM is valid, before returning to the interrupted process.

4.4 HPMC Handler (OS_HPMC)

The OS_HPMC code is established when the operating system is first initialized. It contains an error logging and recovery routine. If PDCE_CHECK determines that the processor is still sufficiently functional, it invokes OS_HPMC. The OS_HPMC code performs error logging plus whatever recovery is possible. The operating system is not required to recover from HPMCs.

The HPMC mechanism is used for error conditions which require immediate attention.

4.4.1 Interface to OS_HPMC

A description of the conditions that need to be met before PDCE_CHECK invokes the OS_HPMC code is given under the OS_HPMC interface description in Section 2.2, PDC Entry Points.

4.4.2 The Role of OS_HPMC

- OS_HPMC must call PDC_PIM (ARG1=0) at least once.
- OS_HPMC uses the information supplied by the PDC_PIM call to determine whether there is storage integrity, and to determine a conservative indication of what recovery actions (if any) need to be taken. A more precise indication of the recovery actions may be obtained from information in system tables, and depends on software convention.
- If it is determined that OS_HPMC can return to the interrupted process, then OS_HPMC must check certain validity fields in PIM to ensure that the CPU state in PIM is valid, before returning to the interrupted process.

4.4.3 Cache Handling During HPMC Processing

A Category B processor executing OS_HPMC in the presence of cache corruption may optionally execute using only non-coherent operations, until OS_HPMC understands the scope of the fault, and knows that it can safely issue coherent operations without the risk of repeated HPMCs. This prevents a cache failure from deadlocking the system. This condition is logged by setting the *rcc* (remote cache coherence check) bit in the Cache Check word in PIM, and operating the cache in a non-coherent mode. A Category B processor may also execute OS_HPMC (in the presence of a cache corruption) in some other SVERSION-dependent manner (e.g., by bypassing the cache).

4.5 LPMC Handler (OS_LPMC)

This code is invoked by PDCE_CHECK to report error conditions that have been completely corrected.

4.5.1 Interface to OS_LPMC

A description of the conditions that need to be met before PDCE_CHECK invokes the OS_LPMC code is given under the OS_LPMC interface description in Section 2.2, PDC Entry Points.

4.5.2 The Role of OS_LPMC

The purpose of OS_LPMC is to only log error information provided in PIM by calling PDC_PIM (ARG1=2) and returning to the interrupted process.

4.6 First-Level Interrupt Handler (OS_FLIH)

The first-level interrupt handler (OS_FLIH) is the routine which handles all HP-PA interruptions. This section considers only external interrupts.

4.6.1 Interface to OS_FLIH

The interface to OS_FLIH is through the Interruption Vector Table. The location of the Interruption Vector Table is specified by the Interruption Vector Address (IVA) register, CR14. The Interruption Vector Table and the processor state upon entry to OS_FLIH are defined in the *Precision Architecture and Instruction Reference Manual*.

4.6.2 External Interrupts

The mechanism used by a module (other than a Type-A Direct I/O module) to interrupt a processor, is the External Interrupt Message (EIM). An EIM contains a 5-bit *group* field that selects one of 32 bits in a processor's EIR register, CR23.

An EIM is directed to the External Interrupt Request (IO_EIR) register of a single processor, or to the IO_EIR register in the LBRS or GBRS (in the case of a broadcast interrupt). To interrupt a single processor, the EIM is sent to the IO_EIR register in that processor's SRS.

Type-A Direct I/O modules use the PATH_INT signal to generate an EIM. This bus signal corresponds to a global broadcast to EIR{3}.

All I/O modules may optionally generate interrupts. A module's IODC indicates whether the module has interrupt capability or not.

Processors can not only receive interrupts, but software executing on a processor can also send an EIM using a store word instruction. By writing to the appropriate address, the EIM can be directed to any single processor, or broadcast to all processors in the system.

Processors receiving a global broadcast on EIR{1} take an interruption by entering OS_FLIH. When OS_FLIH sees EIR{1} set, it clears EIR{1} and enters the remote powerfail warning and recovery (OS_PFW_REMOTE) code.

4.7 Powerfail and Recovery

Precision systems are designed to operate correctly even if their source of primary power fails temporarily. The I/O Architecture provides mechanisms so that software can be warned of an impending power failure and be informed of the power status of system components after power is restored. The recovery strategies depend on memory modules preserving their contents throughout the power failure through the use of secondary power. Software can use information stored in memory to reconstruct its state after power is restored.

The following paragraphs briefly describe ways in which different power failure situations are handled.

In the event of a **normal power failure**, the system is allowed enough time to save its state in memory. After primary power is restored, the system can reestablish its state during powerfail recovery and continue operation.

In the event of an **extended power failure**, the contents of memory are destroyed. In such a case, recovery is not possible. When primary power is finally restored, the system will hard boot.

In the event of **repeated power failures**, the interval between failures is not sufficient to allow powerfail recovery to be completed before the next failure occurs. Recovery algorithms can however be structured to recover in such situations.

A **sudden power failure** is a catastrophic failure, since processor state cannot be preserved. When power is restored, the system will boot.

A **peripheral power failure** is said to have occurred, if an I/O device experiences a power failure while the system is running. As long as its associated module does not lose power, the device driver is responsible for recovery.

In the event of a **total power failure**, recovery can start as soon as the central bus regains power. When all busses regain power, the entire system gets reinitialized.

In the event of a **partial power failure**, it is always possible to recover once all busses regain power by simulating a total power failure; recovery software is established and then every module in the system is given a soft reset.

In general, powerfail recovery software will not be able to resume system operation at the exact place it was interrupted by the power failure. This is because the power failure may have caused the loss of data buffered on modules that do not use secondary power. Thus it is the goal of powerfail recovery software to preserve the integrity of the system through checkpointing and/or restarting incomplete I/O transfers.

PROGRAMMING NOTE

Each system must decide whether to implement all of these steps and hence recover from power failures, or not to recover at all.

The sections that follow define steps necessary to prepare for, and recover from power failures, and offer pointers to illustrative algorithms that software can employ in implementing their recovery strategies. These relate to:

- **Local Powerfail Preparation**

  Power to the central bus is about to be lost. A processor (probably the monarch) disables I/O traffic on the central bus. All processors flush their caches to memory and wait for power to fail.

- **Local Powerfail Recovery**

  Power to the central bus was lost, but the contents of memory remained valid. The monarch processor executes the recovery software pointed to by a known location in Page Zero (MEM_POW_FAIL). The recovery code is protected by a checksum mechanism.

- **Remote Powerfail and Recovery**

  Power on one or more remote busses is about to be lost. An interrupt is sent to the processor(s) in the system on EIR{1}. The operating system stops normal activities and monitors the power status of all remote busses. When stable power returns to the remote busses, the operating system performs recovery and continues normal operation.

PROGRAMMING NOTE

When software is probing bus converters to see if any remote busses have lost power, it should always descend down branches of the tree, checking each node as it goes. This will minimize the occurrence of HPMCs caused by addressing modules that have lost power. However, it is still possible to generate such HPMCs, no matter how careful the software is: The EIR{1} warning of a remote power failure can be delayed in bus converters, and then power can fail at any time.

PROGRAMMING NOTE

When a power failure interrupt occurs, the current state of the processor is saved on the Interrupt Control Stack (ICS). After the ensuing power fail recovery is complete, the state of the processor is restored from the values saved on the ICS. Now a second power failure interrupt can occur before the first power fail recovery is completed. In this case, a second processor state save frame is pushed onto the ICS. In the case of repeated power failures, many power failures follow each other in rapid succession. There is no upper limit to the number of power failures that can occur before the first one is recovered from.

Of course, the size of the ICS is bounded. The probability of overflowing the stack is system dependent. Systems in which pushing a new state save frame for each powerfail warning may exceed the bounds of the ICS can still be accommodated. The key point is that no more than two state save frames need to be stacked, since most of the powerfail recovery procedure is restartable. A global flag can be used to indicate when a state save frame needs to be pushed onto the ICS. When the flag has the value NEST or NORMAL, the processor state must be saved on the ICS. When the flag has the value ROLLBACK, the state need not be saved, because there is already a frame containing the necessary processor state. Clues as to how to use the flag are provided for systems that need it.

4.8 Local Powerfail Warning (OS_PFW_LOCAL)

When power on the local bus begins to fail, the BUS_POW_WARN signal is asserted. As each processor on the local bus observes the assertion of the BUS_POW_WARN signal, it takes a power failure interrupt and invokes the power failure interruption handler (OS_PFW_LOCAL). The interrupt is masked when the PSW I-bit is zero. (One effect of taking the interrupt is that the PSW I-bit is cleared.) Powerfail recovery is not required if an HPMC or TOC occurs during powerfail preparation.

4.8.1 Interface to OS_PFW_LOCAL

The interface to the power failure interruption handler (OS_PFW_LOCAL) is defined in the *Precision Architecture and Instruction Reference Manual*.

4.8.2 The Role of OS_PFW_LOCAL

OS_PFW_LOCAL needs to consider performing the following actions in order to prepare for a power failure and to enable OS_PFR to execute successfully when power returns. (It needs to be noted that during the powerfail sequence, the only PDC calls allowed are PDC_CHASSIS and PDC_POW_FAIL).

1. Disable bus requestorship of all modules on the local bus except for native processors.
2. Save processor state and flush contents of the data cache to memory. Establish correct values for MEM_PF_LEN and MEM_POW_FAIL in Page Zero.
3. Call PDC_POW_FAIL.

4.8.3 Powerfail Considerations in an Multiprocessor Environment

In a multiprocessor environment, software must ensure that all processors have seen the powerfail interrupt, before software executing on one of the processors (probably the monarch) issues the local broadcast flex disable operation. The local broadcast flex disable operation must not be issued until all processors have seen the interrupt, to ensure that no processors have READ or CLEAR operations outstanding in a bus converter.

Software executing on each processor may flush its cache before and/or after the local broadcast flex operation is issued.

After software on all processors have successfully flushed their caches, software on one processor (probably the monarch) writes MEM_POW_FAIL. Software executing on the processor writing MEM_POW_FAIL, must complete the write before calling PDC_POW_FAIL. Software on the other processors call PDC_POW_FAIL after they have completed flushing their caches. All processors must call PDC_POW_FAIL.

Each operating system must also establish a convention as to which processor should write MEM_POW_FAIL.

When BUS_POW_VALID is asserted, all processors execute PDCE_RESET. If memory is still valid, powerfail recovery will be performed.

4.9 Local Powerfail Recovery (OS_PFR)

The local powerfail recovery (OS_PFR) code is invoked by PDCE_RESET on the monarch processor after a power-on, if memory is valid, OS_PFR has been correctly established and is not corrupt.

4.9.1 Interface to OS_PFR

A description of the conditions that need to be met before PDCE_RESET invokes OS_PFR is given under the OS_PFR interface description in Section 2.2, PDC Entry Points.

4.9.2 The Role of OS_PFR

OS_PFR needs to consider performing the following actions if it needs to be successful in ensuring recovery after a power failure.

1. Ensure that the MEM_POW_FAIL vector in Page Zero has a non-zero value only after completion of the power-down sequence and is zero at all other times.
2. Establish the minimum amount of processor state necessary to proceed with powerfail recovery.
3. Call PDC_COPROC to determine which coprocessors are present and functional. Since the state of all coprocessors is HVERSION dependent after calling PDC_COPROC, a state restore sequence of a valid coprocessor state must be used after the PDC_COPROC call to enable use of the coprocessor by the operating system.

4.10 Remote Powerfail Warning (OS_PFW_REMOTE)

When power on a remote bus begins to fail, all bus converters attached to that bus use a global broadcast EIR{1} operation to send a powerfail warning to all other busses. Processors receiving the global broadcast EIR{1} operation take an interruption by entering the first-level interrupt handler (OS_FLIH). When OS_FLIH sees EIR{1} set, it clears EIR{1} and enters OS_PFW_REMOTE code.

4.10.1 Interface to OS_PFW_REMOTE

When OS_PFW_REMOTE is entered, the following conditions are satisfied:

- The processor took an interruption because EIR{1} was set, and entered OS_FLIH.
- OS_FLIH cleared EIR{1}.

4.10.2 The Role of OS_PFW_REMOTE

OS_PFW_REMOTE needs to re-enable interrupts on EIR{1} if it needs to be successful in recovering from power failures on remote busses.

This page intentionally left blank

TABLE OF CONTENTS

This page intentionally left blank