PA-RISC 1.1 I/O Firmware Architecture Reference Specification

Version 1.0

Printed in U.S.A. August 22, 2001

Notice

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THE MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with furnishing, performance, or use of this material.

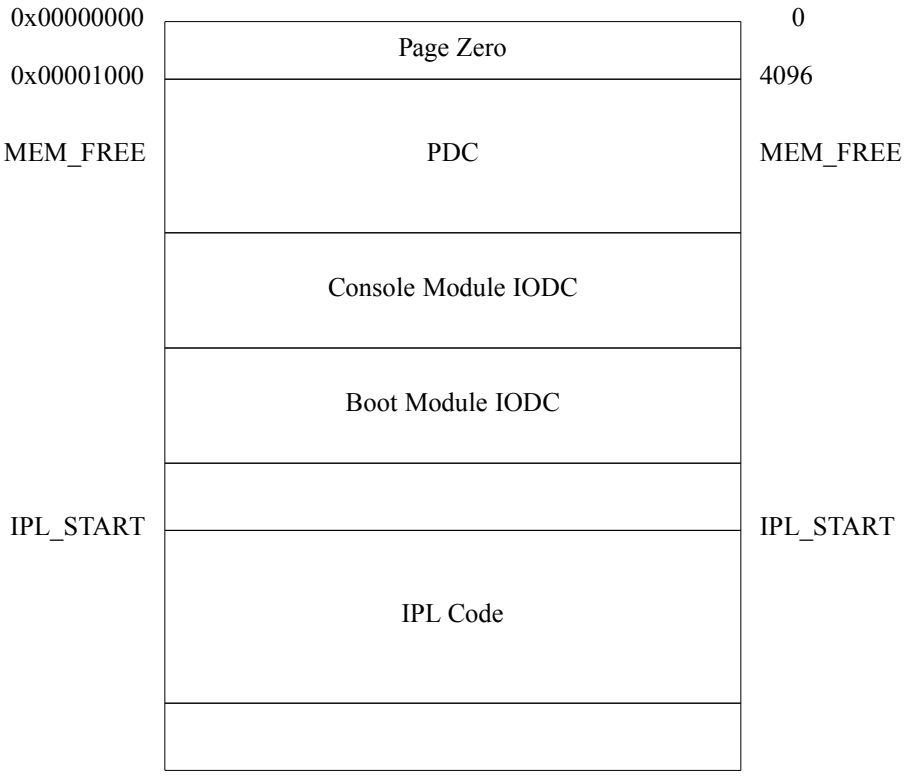
Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

This page intentionally left blank

C. Memory Data Formats

C.1 Data Format of the Initial Memory Module

The Initial Memory Module is unique because of its significance during boot. The format of the Initial Memory Module when PDC invokes IPL is as follows:



Page Zero

The first 4 Kbytes of SPA space on the Initial Memory Module comprise Page Zero, which is used by PDC, IPL, and ISL. The format of Page Zero is described in the next section.

PDC

The PDC area is allocated for use by PDC for code and data. For example, the stack used for calls to PDC and IODC may be located here.

Console Module IODC and Boot Module IODC

The Console Module IODC and Boot Module IODC areas are located in memory following the PDC area and are each a maximum of 32 Kbytes in length. PDC loads ENTRY_INIT and ENTRY_IO for the Console and Boot modules into these areas.

IPL Code

PDC loads the IPL code into memory at address IPL_START. IPL_START must be 4 Kbyte aligned and be less than or equal to 128 Kbytes in size. The format of the code is operating system dependent, but it must include IPL_CHECKSUM. For more details on IPL code, see Section 3.6, Initial Program Load (IPL).

Use of the Initial Memory Module by PDC

PDC is not required to be located in the PDC address space. Instructions and data for PDC procedures and PDC entry points may optionally be stored in memory.

PDC may use any of the processor-dependent regions below MEM_FREE for any HVERSION dependent purpose.

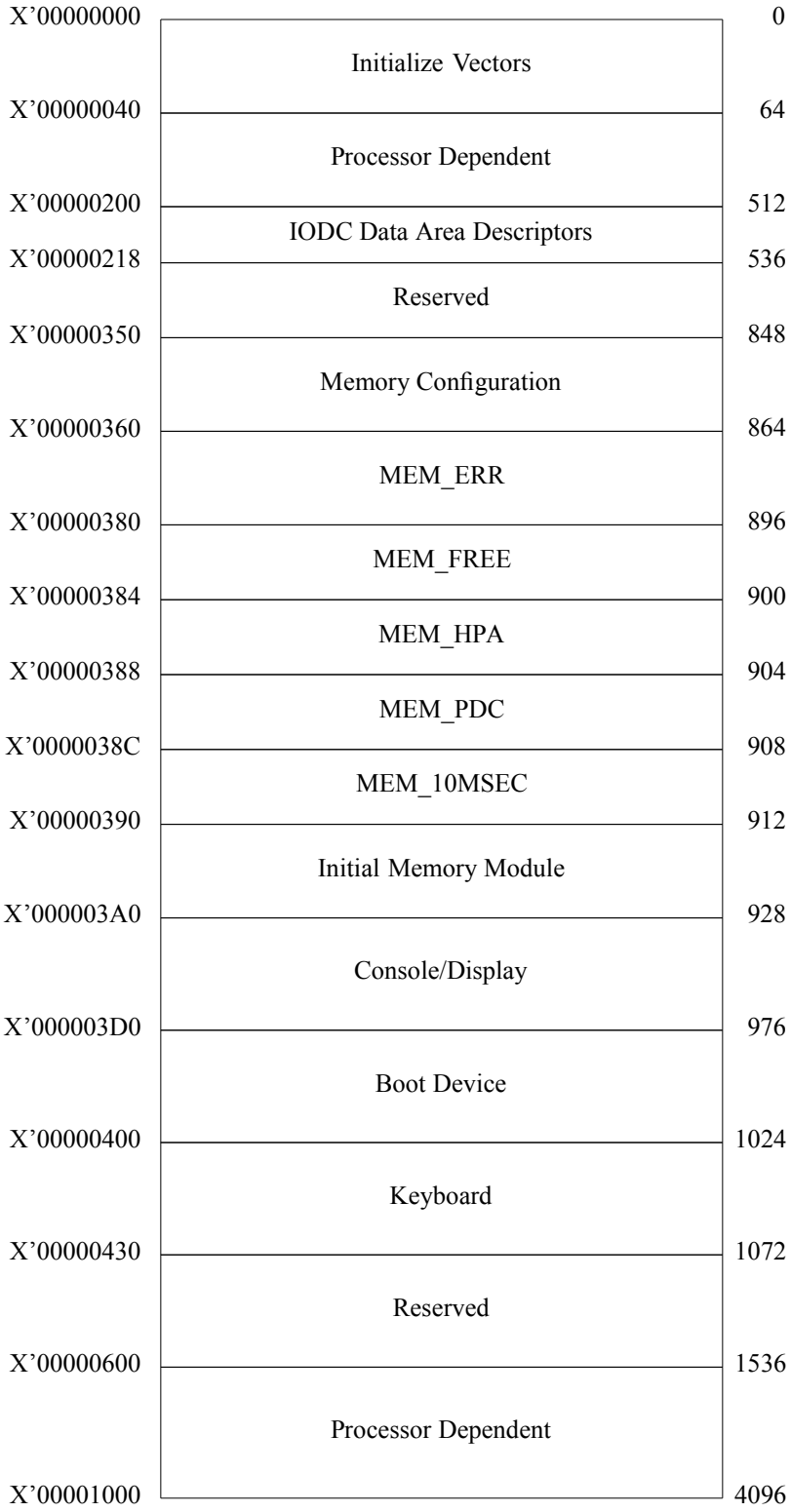
After IPL is launched, PDC must confine its use of memory to the addresses between 0x00001000 and MEM_FREE-1, and to the two areas of Page Zero labeled "Processor Dependent".

PDCE_RESET may be triggered at a time when the Initial Memory Module is not initialized. Therefore, PDCE_RESET must ensure that the Initial Memory Module is initialized before using memory.

SUPPORT NOTE

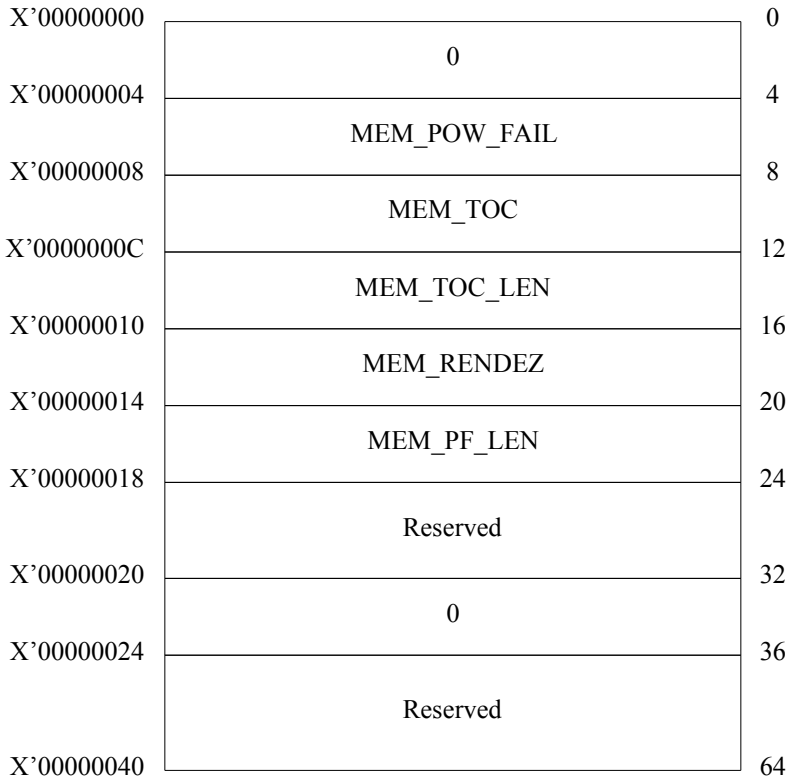
It may increase the supportability of a system if the instructions and data for PDCE_CHECK and PDCE_TOC are stored internal to the processor module rather than in a memory module.

C.2 Data Format of Page Zero



Initialize Vectors

The Initialize Vectors are defined as follows:



The MEM_POW_FAIL vector specifies the location of OS_PFR. MEM_POW_FAIL is valid only when it is nonzero, the contents of memory are valid, and the powerfail checksum is valid.

The MEM_PF_LEN word specifies the number of bytes of code to be checked by the powerfail checksum, starting from the location pointed to by the MEM_POW_FAIL vector. The value of MEM_PF_LEN must be a multiple of 4 and in the range of 0 to 256.

PROGRAMMING NOTE

To minimize the probability of powerfail recovery being attempted when either MEM_POW_FAIL or OS_PFR is invalid, the operating system should make MEM_PF_LEN greater than zero, thus protecting a portion of OS_PFR with the powerfail checksum.

The MEM_TOC vector specifies the location of OS_TOC which is executed after a transfer of control. MEM_TOC is valid only when it is nonzero, the contents of memory are valid, and the TOC checksum is valid.

The MEM_TOC_LEN word specifies the number of bytes of code located in memory at the location pointed to by MEM_TOC. MEM_TOC_LEN must be a multiple of 4 (0 is a legal value).

The MEM_RENDEZ vector specifies the location of OS_RENDEZ which is executed after receiving the rendezvous signal (an interrupt to EIR{0}). MEM_RENDEZ is valid only when it is nonzero and the contents of memory are valid.

The word at byte address 0x00000020 is zero so that, when an HPMC is taken with an IVA of 0x00000000, the processor will halt, rather than attempt to execute an undefined HPMC handler. The remaining vectors are reserved, and required to be 0 if the contents of memory are valid.

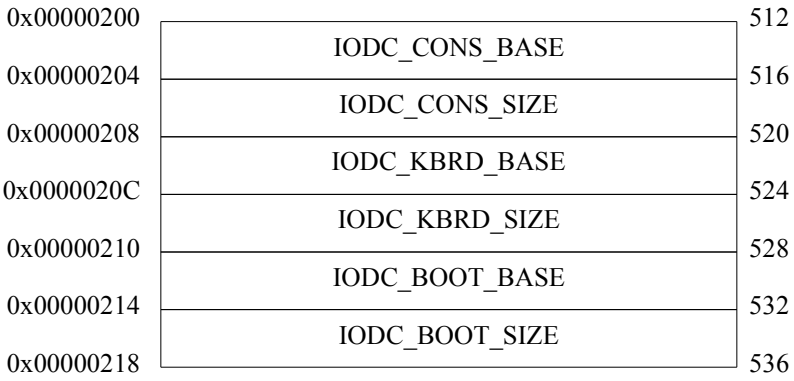
Processor Dependent

The Processor Dependent areas are allocated for HVERSION-dependent purposes. In particular, an implementation may use these areas for Processor Internal Memory (PIM). If these areas contain PIM for a processor, PDC must not clear the areas during soft boot since the saved data would be destroyed.

IODC Data Descriptors

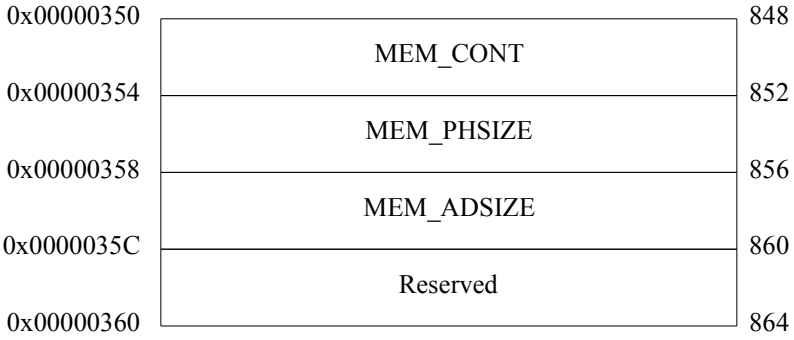
The IODC Data Descriptors are used to specify which areas in the IODC portion of low memory can be used for Console, Keyboard, or BOOT IODC.

Each area may be allocated using PDC_ALLOC. One allocated, the Console and Keyboard areas are static and should not be changed. The boot area can be freed by zeroing IODC_BOOT_BASE and IODC_BOOT_SIZE, then calling PDC_ALLOC to allocate the area again. PDC_ALLOC should not be called if the values in the corresponding BASE and SIZE field are not zero.



Memory Configuration

The memory configuration established by PDC is specified by four words, as follows:



MEM_CONT specifies the number of bytes of contiguous memory, starting at address zero, which were configured by PDC. Utilities which cannot handle noncontiguous memory use this value as an upper address bound.

MEM_PHSIZE specifies the total amount of physical memory which was configured by PDC and is available to a utility which can handle noncontiguous memory. MEM_CONT and MEM_PHSIZE are equal in a system with contiguous memory.

MEM_ADSIZE is the total amount of memory SPA space which was configured by PDC; any utility which wishes to configure additional memory must not use SPAs less than this value.

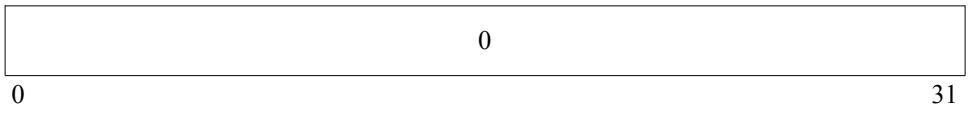
PROGRAMMING NOTE

If software finds MEM_CONT, MEM_PHSIZE, and MEM_ADSIZE set to zero (which is allowed only to support previous implementations), it must default to using the *max_mem* word in the Initial Memory Module.

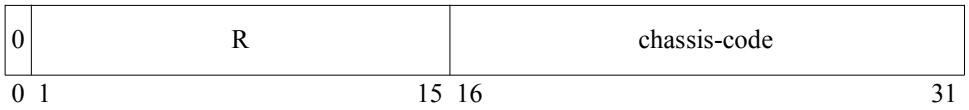
MEM_ERR

An eight word block is used to report errors detected during boot to software, one error per word. A value of zero indicates that no error is reported and a nonzero value indicates that either an advisory chassis code error or a memory failure has been logged. When logging errors in MEM_ERR, the logging of memory failures take precedence over the logging of advisory chassis code errors. If there are more than eight memory failures to be logged during boot, the monarch processor must halt. Implementations must never log chassis code errors which are fatal in nature. Each word uses one of the three following formats:

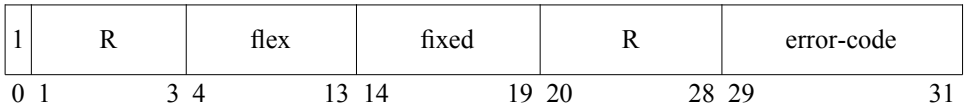
a. *all zero* format - indicating no error



b. *chassis code* format - indicating advisory errors



c. *error code* format - indicating memory failures



The *flex* and *fixed* fields are part of the HPA of the failing memory module.

Assignments to the *error-code* field are as follows:

Value	Description
0	hard boot correctable error
1	hard boot uncorrectable error
2	soft boot correctable error
3	soft boot uncorrectable error
4	ENTRY_TEST error: module is not usable
5	ENTRY_TEST error: module is fully functional at degraded performance
6-7	reserved

MEM_FREE

MEM_FREE points to the first available location that the operating system can overwrite. The maximum allowed value of MEM_FREE is 64 Kbytes. PDC can use the MEM_FREE location for HVERSION-dependent purposes until it gives control to IPL.

PDC may optionally use memory locations greater than the value of MEM_FREE during soft boot. The maximum memory location that can be altered by PDC during soft boot is:

MEM_FREE + 32 Kbytes(Console Module IODC) + 32 Kbytes(Boot Module IODC) - 1

Memory past this location must not be written to by PDC during a soft boot, except to load IPL.

ENGINEERING NOTE

Any code or data that PDC places beyond the value of MEM_FREE may be overwritten by IODC during soft boot.

MEM_HPA

The MEM_HPA word contains the HPA of the monarch processor. The MEM_HPA word must be established by PDCE_RESET before any IODC calls and before the IPL code is invoked.

MEM_PDC

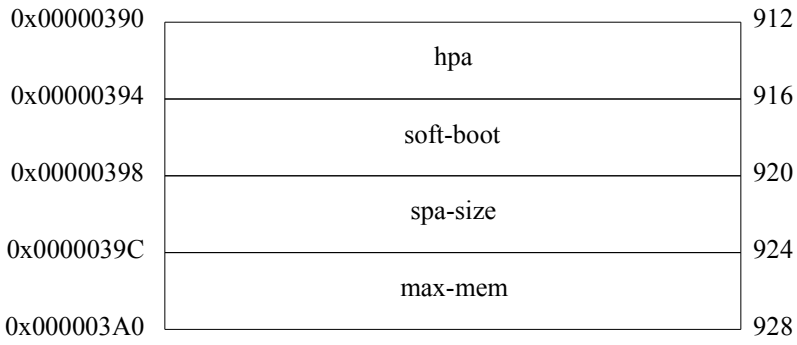
The MEM_PDC word contains the address of PDCE_PROC for the monarch processor. PDCE_RESET must establish MEM_PDC after establishing PDCE_PROC, but before calling any IODC entry point, invoking IPL, or invoking OS_PFR.

MEM_10MSEC

The MEM_10MSEC word contains a calibration factor for the Interval Timer (CR16) of the monarch processor. This calibration factor specifies the number of clock ticks in 10 msec. The MEM_10MSEC word must be established before any IODC calls and before the IPL code is invoked.

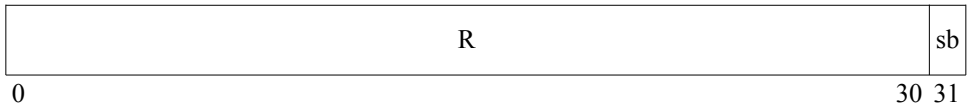
Initial Memory Module

The configuration of the Initial Memory Module is specified by four words, as follows:



The hpa word is the HPA of the Initial Memory Module.

The format of the soft-boot word is as follows:



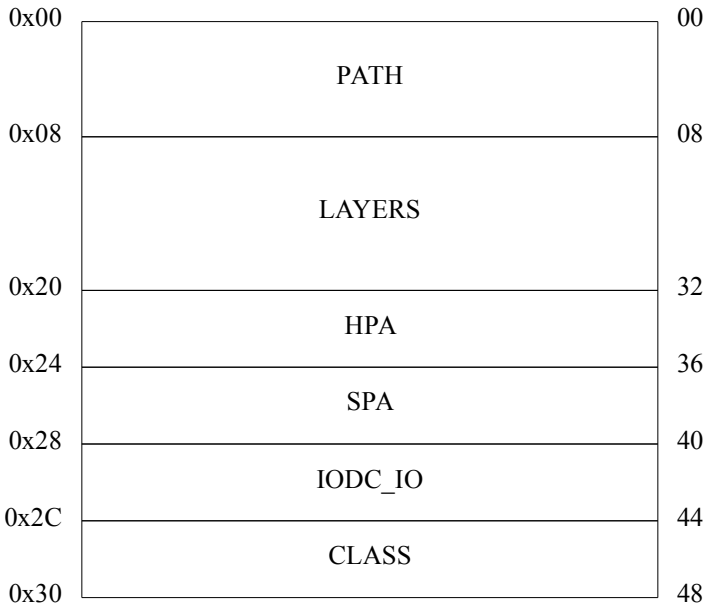
The *sb* bit is 0 if the most recent boot was a hard boot. The *sb* bit is 1 if the most recent boot was a soft boot. If the processor does not distinguish between types of boot, *sb* must be set to 0.

spa-size specifies the size of the IMM's SPA space in bytes.

max-mem specifies the size of implemented memory in bytes.

Console/Display, Boot Device, and Keyboard

Console/Display, Boot Device, and Keyboard are identical structures of 48 bytes each. The format is as follows (the offsets are in bytes from the start of the structure):

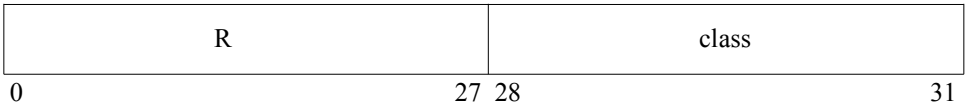


The PATH and LAYERS structures are identical to the corresponding structures of Primary Boot Path in Stable Storage.

HPA specifies the base of the module's hard physical address space. SPA specifies the base of the module's soft physical address space. The SPA word is 0 if no SPA exists for the module. If the I/O configuration is changed, software must update the HPA and SPA words to the correct values.

The IODC_IO word gives the memory address of the module's ENTRY_IO IODC entry point. This pointer must be 0 if the module's ENTRY_IO is not present in memory.

The CLASS word is formatted as follows:



The *class* field specifies the I/O device class, in terms of its physical properties, as follows:

Value	Name	Description
0	CL_NULL	Invalid
1	CL_RANDOM	Random access media (as in disk)
2	CL_SEQU	Sequential record access media (as in tape)
3 - 6	Reserved	Reserved
7	CL_DUPLEX	Full duplex point-point communication (as in RS-232)
8	CL_KEYBD	Half-duplex console (Keyboard In)
9	CL_DISPL	Half-duplex console (Display Out)
10	CL_FC	FiberChannel access media
11 - 15	Reserved	Reserved

Some of the boot device classes (e.g., CL_DUPLEX) can be used as either a console device or boot device. Since the device class is insufficient to distinguish a boot device from a console terminal, only boot devices with class CL_RANDOM or CL_SEQU should be used in an autosearch path.