PowerPC SMP binding to:

IEEE Std 1275-1994

Standard for Boot

(Initialization, Configuration)

Firmware

Revision: 0.04 DRAFT

Date: July 10, 1995

1. Overview

This document specifies the application of *IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware, Core Practices and Requirements* to PowerPC symmetrical multiprocessor (SMP) systems. An implementation of Open Firmware for these PowerPC systems *shall* also implement the core requirements as defined in [1], the PowerPC Processor-specific extensions described in [2], and the PowerPC platform specific defined in [3] or [4] depending on whether the system is to be PowerPC Reference Platform compliant or Hardware Reference Platform (HRP) compliant.

While this document addresses the official PowerPC architecture [2], the name "PowerPC" only requires compliance to Book I. The descriptions that follow, and the relevant sections describing translation features for this binding, assume that the system's PowerPC processor(s) implement the entire set of Books I-III. Some "PowerPC" processors may implement different Book II-III features; such processors may need a variant of this binding describing the differences to the mapping functions, etc.

This document defines the binding to PowerPC processors that use 32-bit addressing. Since the minimum cell size of Open Firmware is 32 bits, only one cell is necessary to represent addresses of processor bus devices; hence, the value of "`#address-cells`" for `/` *shall* be 1.

> **Note: 64-bit processors can follow the specifications contained herein as long as all addresses relevant to Open Firmware are kept within the first 4 GB; i.e., the upper 32 bits of 64-bit addresses are assumed to contain 0s.**

1.1. References

This binding document shall be used in conjunction with the following publications. When the following standards are superseded by an approved revision, the revision *shall* apply.

[1] *IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware, Core Practices and Requirements*.

[2] *PowerPC processor binding to: IEEE Std 1275-1994, Standard for Boot (Initialization, Configuration) Firmware.*

[3] *PowerPC Reference Platform binding to: IEEE Std 1275-1994, Standard for Boot (Initialization, Configuration) Firmware.*

[4] *PowerPC HRP binding to: IEEE Std 1275-1994, Standard for Boot (Initialization, Configuration) Firmware.*

[5] *System V Application Binary Interface*, published by UNIX System Laboratories. This document describes the generic architecture of the ELF (Executable and Linking Format) object file format.

[6] *MS-DOS Programmer's Reference*, published by Microsoft. This document describes the MS-DOS partition, directory and FAT formats used by `disk-label` support package.

[7] *Peering Inside the PE: A Tour of the Win32 Portable Executable File Format*, found in the March, 1994 issue of *Microsoft Systems Journal*.

[8] *Bootstrap Protocol*, Internet RFC 951; see also RFC 1532.

[9] *ISO-9660, Information processing -- Volume and file structure of CD-ROM for information interchange*, published by International Organization for Standardization.

[10] *System V Application Binary Interface, PowerPC Processor Supplement*, Sunsoft. This document defines the PowerPC specific ABI for System V and also gives details on the PowerPC ELF format.

[11] *Device Support Extensions to IEEE 1275-1994 Standard for Boot (Initialization, Configuration) Firmware*.

[12] *IEEE 1275 extensions for Multi-color Text and Graphics Support*.

1.2. Terms

This standard uses technical terms as they are defined in the documents cited in “References”, plus the following terms:

core, core specification: refers to *IEEE Std 1275-1994 Standard for Boot (Initialization, Configuration) Firmware, Core Practices and Requirement.*

2. The Device Tree

Open Firmware requires that the multiple instances of any device that appears more than once in the device tree must be distinguishable by means of their “reg” properties. The “reg” property must express the “address” of each node relative to its parent “bus”. Furthermore, the core specification says that the root node of the device tree usually represents the “main physical bus” of the system. Thus, if processors are not directly addressable on the main physical bus, as is expected to be the case on many/most PowerPC-based systems, the CPU nodes on such systems may not be children of the root node but must instead be children of a pseudo-device node. In this case, the name of the pseudo-device node, which will usually be a child of the root node, shall be “cpus”.

> **Note: The properties of a CPU node are already defined in the PowerPC Processor binding document. The only change for multiprocessor (MP) systems is that there may be multiple CPU nodes, with a special “cpus” pseudo-bus above it if the CPUs aren't addressable on some parent “real” bus. Other properties of the CPU nodes shall conform with the requirement in the PowerPC Processor binding document.**

The “cpus” node shall have one child node of device-type “cpu” for each processor. The value of the “reg” property of each CPU node shall be a prop-encoded-array, an integer encoded as with encode-int; its textual representation shall be the hexadecimal representation of the integer. A CPU with a dedicated (non-shared) cache shall have an “l2-cache” property in the CPU node itself; the “l2-cache” property for a shared L2 cache shall appear in the “cpus” node. It also is required to publish the i-handle of the “running” CPU, in the property “cpu” in the /chosen device node.

> **Note: With respect to the reg property, the specification of encoded int is true only for the case of the “cpus” pseudo-bus because, if anyone implements a PowerPC system (non-HRP) that puts the CPUs on a real bus, e.g., the “main physical address bus” represented by the root node, the reg property would be different (i.e., in the form required by that bus).**

3. Initialization

Open Firmware shall select one processor, using an algorithm of its choice, to be the “master” processor, which performs the role of the single processor on a uniprocessor system, either booting the client or providing the user interface. Open Firmware shall place all the remaining processors into stopped state, a state in which the processor does not perform Open Firmware or client functions and does not interfere with the operation of the master processor. A processor in stopped state remains in that state unless and until an executing client starts it using the START-CPU client service defined below.

4. Client Interface Services

The following client services are added for SMP support on PowerPC systems. These interfaces make the client program responsible for any Inter-CPU communication needed for these interfaces. The rationale for this is to architecturally separate the Inter-CPU communication mechanism of the firmware from the client program and vice versa.

4.1. START-CPU

start-cpu (nodeid, pc, arg --) Start CPU

- The nodeid is the phandle of a node whose device-type is “cpu”.

- START-CPU arranges for the CPU identified by *phandle* to begin executing client code at the real address given by the *pc* input with an argument, arg, passed in register r3. When it begins execution, the started processor shall be in the endian mode of the client, and in real (not translated) addressing mode. The contents of registers other than r3 are indeterminate.

Note: We shall not allow a processor to START-CPU itself, effectively restarting with a new pc and abandoning the only client thread. A jump or branch instruction shall be used instead to achieve the objective.

- START-CPU permits more than one processor to run at the same time, enabling multi-threaded client execution. In general, an Open Firmware client shall avoid multi-threaded operation within Open Firmware. Usually, this means that client threads running on different CPUs must use mutual exclusion to prevent more than one processor from making client service requests at any one time. The exceptions are that a client thread may invoke either the STOP-SELF or IDLE-SELF client services defined below at any time.

Note: The results are undefined if the CPU identified by *phandle* has already been started (e.g it is already running and has not exited) or *phandle* is not a valid package handle of a CPU device node.

4.2. STOP-SELF

stop-self(--)Stop the CPU calling this client interface.

- Open Firmware places the processor on which the caller was running into the stopped state.
- The client program is not-resumable.

Note: When an MP client program exists, one CPU invokes the EXIT client interface service, the others invoke the STOP-SELF service.

4.3. IDLE-SELF

idle-self(--)*Idle* the CPU calling this client interface.

- Open Firmware places the CPU on which this service was invoked into an 'idle' state, saving the *current state* of the client program, so that the client program may be resumed.
- A processor in idle state can be resumed using RESUME-CPU service defined below or restarted using START-CPU. If the processor is resumed, it executes a normal return to the client, as if its call to IDLE-SELF had just completed.

Note: When a client program wants to enter the firmware user interface, one CPU invokes the ENTER client interface service, the others invoke the IDLE-SELF service. The rational is that the user interface may affect the machine state in any way that it desires, therefore the client shall not depend on it.

4.4. RESUME-CPU

resume-cpu(nodeid --)Resume an *idled* CPU.

- The nodeid is the phandle of a CPU node in idle state.
- RESUME-CPU arranges for that CPU to restore the CPU’s state as saved by IDLE-SELF and begin return to the client, completing the idle-self client service call that placed the CPU into idle state.
- The results are undefined if the CPU identified by *phandle* is not in an *idle* state by a previous call to the IDLE-SELF client interface service.

Note: When the client program is resumed via the GO (or similar) user interface command, the client program is resumed on the CPU which called the ENTER service; the client program is responsible for calling the RESUME-CPU service to resume other idled CPU's, if that is the desired client program behavior.

5. Breakpoints

If the breakpoint is taken by the firmware, without the client program's assistance, the other CPUs will continue to run in the client program. The client program may field the breakpoint 'trap' or 'vector' and idle the other CPUs before entering the PROM. The platform binding document has to specify how this is done to avoid loss of state at breakpoint time.

6. Locking Mechanisms

The firmware is a single threaded program, from the client program's point of view. Only the IDLE-SELF, STOP-SELF, ENTER and EXIT client interfaces may be invoked simultaneously on different CPUs. Furthermore, only a single CPU may invoke the ENTER or EXIT client interface at any one time. The other CPUs must use the IDLE-SELF or STOP-SELF client interface service.

> **Note: The results are undefined if the client program invokes client interface services including breakpoint traps (other than the ENTER/EXIT STOP-SELF/IDLE-SELF case listed above) simultaneously on more than a single CPU.**

> **Note: Since locking mechanisms are subject to client program policy, the client program is responsible for implementing any necessary mechanism to insure that it adheres to this policy. Further, the client program should disable any pre-emption mechanism before calling a client interface service to avoid rescheduling a thread of execution in the firmware on a different CPU if such a mechanism exists in the client program.**