PowerPC Reference Platform binding to:

IEEE Std 1275-1994

Standard for Boot

(Initialization, Configuration)

Firmware

Revision: 0.02 DRAFT

Date: July 3, 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 Reference Platform Compliant computer systems, including practices for client program interface and data formats. An implementation of Open Firmware for a PowerPC Reference Platform Compliant system *shall* implement the core requirements as defined in [1], the PowerPC Processor-specific extensions described in [2] and the PowerPC Reference Platform specific extensions described in this binding.

2. References and Terms

2.1. References

This standard 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 When the following standards are superseded by an approved revision, the revision *shall* apply.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] *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.

[4] *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.

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

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

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

[8] *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.

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

[10]Open Firmware Recommended Practice: 16-color Text Extension.

[11] *Open Firmware Recommended Practice: Graphics Extension.*

2.2. 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 Requirements*

ELF: Executable and Linking Format. A binary object file format defined by [3][8] that is used to represent *client programs* in Open Firmware for PowerPC.

FDISK: Refers to the boot-record and partition table format used by MS-DOS, as defined in [4].

Open Firmware: The firmware architecture defined by the core specification or, when used as an adjective, a software component compliant with the core specification.

PE: Portable Executable. A binary object file format defined by [5]; this format is used by Microsoft's NT operating system.

Real-Mode: The mode in which Open Firmware and its client are running with translation disabled; all addresses passed between the client and Open Firmware are real (i.e., hardware) addresses.

Suspend: A form of Power Management characterized by a fast recovery to full operation. Typically, system memory will not be powered off while in the suspend state.

Virtual-Mode: The mode in which Open Firmware and its client share a single virtual address space, and address translation is enabled; all addresses passed between the client and Open Firmware are virtual (translated) addresses.

3. Packages

This section describes the PowerPC Reference Platform-specific requirements of Open Firmware packages.

3.1. "`disk-label`" Support Package

This section describes the partition formats and the formats of client program images that the `disk-label` support package for PowerPC *shall* support; an implementation *may* support additional mechanisms, in an implementation-specific manner.

The process of loading and executing a client program is described in two stages. The first stage determines what file to read into memory. This is done by locating a file from the boot device, usually by means of a name lookup within a directory contained within a disk "partition". The second stage examines the front portion (header) of the image for "well-known" program formats. When the format of the image has been determined, the loading is completed in a manner determined by that format.

The name of the file (and, the partition in which it is contained) can be explicitly specified by the user via the **`load`** or **`boot`** command, or can be implicitly specified by the value of the "`boot-device`" property of the "`/options`" node. The filename is the `ARGUMENTS` portion of the final `COMPONENT` of the `PATH_NAME`, as described in section 4.3.1 of [1].

The syntax for explicit filename specification is as follows:

```
[n][,filename]
```

where `n` is the partition number to be used and `filename` is the name of a file within that partition. If `n` is omitted, the default partition (as determined by the partition format) is used. If `filename` is omitted, the default filename (i.e., the filename component of the "`boot-device`" path-name) is used. Partition number 0 refers to the entire device; this definition is independent of the existence of the device partition information.

3.1.1. Partition formats

Open Firmware shall implement the algorithm described as follow:

`Algorithm locating boot file`

```
read sector 0 (bootsector)
if last 2 bytes of sector are 0AA55h (little-endian)
     if bsMedia == 0F8h \ FDISK partition on hard drive
          if an explicit partition has been requested
               select partition number n
          else
               select bootable partition (80h in peBootable field)
          use directory of the selected partition to locate file
     else (non-partitioned)
          use FAT-12/FAT-16 directory to locate file
else
```

```
read sector 16.
if a valid ISO-9660 directory is found
    locate the file, using the ISO-9660 directory.
else
    FAIL, in an implementation-specific manner.
```

This algorithm can be used to locate the correct file and/or load image from the specified device. The boot device is selected as described in 7.4.3.2 of [1]. A filename can be explicitly given as the arguments field of the *device-specifier* (i.e., the field following the ':' of the last path component). Other formats *may* be recognized in an implementation-specific manner.

Although the above algorithm works independent of the device type, the following formats are strongly recommended for devices for portability.

Floppy Disk

1.44/2.88 MB, MFM floppy disks should be in FAT-12 format, as described in [4].

Hard Disk

Hard Disks should have an FDISK partition map, as described in [4].

> **Note: since the bootsector is used to contain boot program for floppies and the FDISK partition map for hard disks, the** `"disk-label"` **package must use the value of the** `bsMedia` **byte (located at offset** `15h`**) to determine whether a partition map is present. If the value is** `0F8h`**, it indicates a hard disk and a partition map should be present in the bootsector; any other value indicates a floppy disk.**

CD-ROM

CD-ROMs should be formatted according to ISO-9660, as described in [7].

3.1.2. Program-image formats.

Open Firmware must recognize a client program that is formatted as either ELF [3][8] or PE [5]. PE format support is provided only for booting NT; all other clients *shall* use ELF. Other formats *may* be handled in an implementation-specific manner.

After locating the file, Open Firmware reads the image into memory at the location specified by the **load-base** Configuration Variable. Then, Open Firmware must perform the following procedure to prepare the image for execution.

init-program.

```
examine the header of the image.
set restart? false
if the image is in ELF format
    if the EI_DATA field does not match little-endian?
        set little-endian? appropriately.
        set restart? true
    locate the PowerPC Note Section
    if the Note Section's descriptor is not correct
        set Configuration Variables appropriately
        set restart? true
    if restart?
        restart the system, possibly by executing reset-all
    else
        move and/or relocate the ELF image.
else
if the file is in PE format
    if little-endian? is false
        set little-endian? to true.
        restart the system, possibly by executing reset-all
```

```
        else
            move and/or relocate the PE image.
    else
        FAIL, in an implementation -specific manner.
```

3.2. "`obp-tftp`" Support Package

The "`obp-tftp`" Support Package of an Open Firmware implementation (used to **`load`** from "`network`" devices) *shall* support the BOOTP protocol, as described in [6] and *shall* not require the server to support any vendor extensions.

4. Properties

This section describes the standard properties of PowerPC Reference Platform Open Firmware implementation.

4.1. "/openprom" node property

Open Firmware must implement the "halt-address" property under the "/openprom" node. The property value describes the location of physical memory where the firmware saves the halt procedure image. Halt procedure is the self-contained firmware callback service for system power-off or reboot. Halt procedure is discussed further in Section 5.1.

"`halt-address`"

prop-encoded-array: address, length

address, encoded as with **encode-int**, is the real mode address of the halt procedure, and length, encoded as with **encode-int**, is the physical memory size for the halt procedure. If the halt procedure is located within read-write memory, length is the number of bytes of memory, encoded as with encode-int, occupied by the halt procedure. If the halt procedure is located within read-only memory, length is 0.

Note: "read-only memory" refers to memory that cannot be written in nomal operation; ROM or FLASH ROM is considered to be "read-only"; while RAM, whether or not it is write-protected via an MMU, is considered to be "read-write".

The read-write memory, if any, that the halt procedure occupies must not be included within the "available" property of the "/memory" node.

5. Client Program Requirements

5.1. Halt callback

The halt callback allows a client program to invoke limited firmware services for turning-off or rebooting the machine. The halt procedure shall not return control to the client program that invoked it. Consequently, the halt procedure is not required to preserve any machine state on behalf of its caller.

PowerPC Reference Platform Open Firmware must implement the "`halt-address`" property within the "/openprom" node to support the halt callback. Detailed discussion about this property is presented in Section 4.1.

5.1.1. Halt procedure calling conventions

When control is transferred to the halt procedure, the system must be in the endian mode that was in effect when the firmware last had control of the system. The client program must establish the real mode address translation for the firmware.

5.1.2. Halt procedure argument

The client program must set GPR3 to point to the halt argument. The halt argument is the address of a null-terminated text string. In the following discussion, the string is called the "halt string".

5.1.2.1. Power-off Service

If the halt string is "power-off", the firmware must turn off the system power to the extent possible. If the firmware cannot turn off the system power, perhaps due to lack of hardware capability, the result is undefined, and the firmware must not try to return control to the client program or reboot the system. The suggested behavior in this case is to enter a firmware interactive mode, if available.

5.1.2.2. Suspend Service

If the halt string is "suspend", the firmware shall preserve the RAM image across a hardware suspend operation in the following manner:

1. The Firmware shall not alter RAM that is not devoted to firmware use (see System State at Call below).
2. If the firmware encounters an error or the hardware does not support suspend, the halt procedure shall set the appropriate return code (see System State at Return below) and return to the caller.
3. The firmware shall record suspend restoration parameters in a system-dependent manner.
4. The firmware shall set the system power state to suspend in a system-dependent manner (this usually means powering down everything except the memory sub-system and its refresh).
5. When the system is subsequently powered up, the firmware performs the following steps:
    - Reads the suspend restoration parameters.
    - Initializes the system hardware as on IPL without disturbing the preserved RAM image
    - Does not load the standard client program (so as not to disturb the preserved RAM image).
    - Resets the suspend restoration parameters.
    - Returns to the caller of suspend (see system State at return below).

The System State at the Call to "suspend" shall be as follows:

Memory:

The memory range defined by the "**real-base**" and "**real-size**" configuration variables is undefined but available for use by Open Firmware.

Master Processor:

GPR3 shall hold the starting address of the null-terminated text string "suspend".

Slave Processors:

In an SMP system only one processor shall call the Halt procedure with a halt string of "suspend", all other processors shall call the halt procedure with the halt string "slave". In this case, the halt procedure shall execute a program which does not change the state of memory.

I/O Sub-system:

All elements of the I/O sub-system shall be in a safe state. In a safe state, they shall not be transferring data to or from memory nor shall they cause an interrupt to any processor.

Interrupt Sub-system:

The processor's External Interrupt Enable bit (MSR[EE]) shall be 0 prior to calling the halt procedure.

The System State at a Return from "suspend" shall be as follows:

Memory:

The state of the memory locations, exclusive of the range defined by the "**real-base**" and "**real-size**" configuration variables, shall be the same as the state they were in at the time of the call to the Halt procedure.

Master Processor:

The state of the master processor shall be as defined for the "initial program state" as specified in [2] with the following exceptions: The processor will start executing at the address which was in the master processor's Link Register at the time the Halt procedure was called. GPR3 shall contain the return code (see below).

Slave Processors:

The state of the slave processors shall be as follows: The MSR shall be set to the state specified for "initial program state" as specified in [2]. All non-architected processor dependent registers shall be set to the system default value. All other architected registers are undefined. The processor shall start executing at the address which was in the slave processor's link register when the Halt procedure was called. GPR3 shall contain the return code 0.

I/O Sub-system:

All the elements of the I/O sub-system shall be initialized to a safe state. The device tree shall be re-established as on IPL.

Interrupt Sub-system:

The processor's External Interrupt Enable bit (MSR[EE]) shall be 0.

Return Codes:

0: Restored from suspend.

-1: Hardware does not support suspend power state.

5.1.2.3. Other Halt Services

The firmware may recognize strings other than "power-off" or "suspend" in a system-dependent manner. If the halt string is not one of the recognized commands, the firmware must reboot the system. During the system reset and the firmware restart, the firmware must preserve the halt string. Then, the firmware must evaluate the string as if "auto-boot?" were true and "boot-command" were set to that string, without altering the values of the "auto-boot?" and "boot-command" configuration variables. If the firmware includes an Open Firmware user interface, the string can be any valid user interface command string. Otherwise, the firmware shall interpret the string as shown in Table 1. Halt arguments.

Table 1. Halt arguments

String value	Meaning
boot	Load and execute the default client program.
<empty string>*	If the firmware has an interactive mode, enter that mode. Otherwise the result is undefined, except that the firmware shall not perform the "boot" action.
<any other string>	Behavior is system-dependent

* Note: <empty string> means a valid address that points to a null character. An address value of 0 is not an empty string.

6. Extensions for PowerPC Reference Platform systems

This section describes the properties, methods, and device subtrees that are applicable to devices required by the PowerPC Reference Platform architecture. It is strongly recommended that other platforms follow these definitions for the corresponding devices.

6.1. Display devices

Display device packages (i.e.., `device_type = "display"`) for PowerPC Reference Platform systems should include in their implementation all the properties and methods called for in [10] and [11].

6.2. Device support

Open Firmware implementations for PowerPC Reference Platform systems must implement those device types from the IEEE 1275 Device Support Extensions document [9] that are appropriate to the hardware present.

6.3. Conventions for devices on ISA and SCSI buses

This section defines the naming and device type conventions for typical devices on ISA and SCSI buses. The following lists are the values of the "name" and "device-type" properties of the devices on an ISA bus:

name	device_type
8042	
kbd	"keyboard"
mouse	"mouse"
floppy	"block"
com	"serial"
timer	"timer"
lpt	"parallel"
ide	"block"
nvram	"nvram"
rtc	"rtc"

Note: The "kbd" and "mouse" names are indented to show that they are the child nodes of the 8042 node.

Some systems use an I/O controller, often called a super I/O chip, which provides control functions of multiple I/O devices. When a system uses a super I/O chip, a device node representing the super I/O chip itself need not exist. Instead, the device nodes of the devices attached to the super I/O chip may be direct children of the bus node representing the bus to which the super I/O chip is attached.

The following are the values of the name and device-type properties of the devices on SCSI bus:

name	device_type
scsi	"scsi"
disk	"block"
tape	"byte"

Note: SCSI controller is considered a bus device in the device tree for PowerPC Open Firmware. The "disk" and "tape" names are indented to show that they are the child nodes of the scsi node.

It is strongly recommended that the "compatible" property be implemented for ISA and SCSI bus devices to help operating systems find appropriate device drivers for these devices.

6.4. /aliases node properties

An implementation of Open Firmware for the PowerPC Reference Platform shall provide the following aliases under `"/aliases"` node if an applicable device exists:

```
disk
tape
cdrom
keyboard
screen
scsi
com1
com2
floppy
net
```