CHRP[TM] VGA Display Device Binding

to:

IEEE 1275-1994

Standard for Boot

(Initialization, Configuration)

Firmware

Revision: 1.0 Unapproved DRAFT

Date: May 6, 1996

Table of Contents

1. Purpose of this Device Binding

This document specifies the application of Open Firmware to PowerPC Common Hardware Reference Platform (CHRP™) VGA devices, including device-specific requirements and practices for initialization, properties, and methods.

2. Revision History

Revision 1.0 Unapproved DRAFTMay 6, 1996	Initial revision. Jordan Brown, Sunsoft and John Kingman, IBM editors

3. References

This Open Firmware System binding 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 When the following standards are superseded by an approved revision, the revision *shall* apply.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] *Core Errata, IEEE P1275.7/D4.*

[3] *PCI Bus binding to: IEEE Std 1275-1994, Standard for Boot (Initialization, Configuration) Firmware* Version 1.7.

[4] *PowerPC Microprocessor Common Hardware Reference Platform: I/O Device Reference*. This document describes the PowerPC Common Hardware Reference Platform (CHRP™) System Standard I/O Devices; hardware registers, register locations, and hardware attributes.

[5] *Open Firmware Recommended Practice: Generic Names*.

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

4. Definition of Terms

5. Device Characteristics (Informative)

The Video Graphics Array (VGA) function includes a video buffer, a video digital-to-analog converter (DAC), a CRT controller, a sequencer unit, a graphics controller, and an attribute controller.

Video memory (buffer) consists of at least 256KB; its use and mapping depend on the mode selected. It is configured as four 64KB memory maps:

The *video DAC* contains the color palette that is used to convert the video data into the video signal that is sent to the display. Three analog signals (red, green, and blue) are output from the DAC.

The *CRT controller* generates horizontal and vertical synchronization signal timings, addressing for the regenerative buffer, cursor and underline timings, and refresh addressing for the video memory.

The *sequencer* generates basic memory timings for the video memory and the character clock for controlling regenerative buffer fetches. It allows the system to access memory during active display intervals by periodically inserting dedicated system microprocessor memory cycles between the display memory cycles. Map mask registers in the sequencer are available to protect entire memory maps from being changed.

The *graphics controller* is the interface between the video memory and the attribute controller during active display times, and between video memory and the system microprocessor during memory accesses.

During active display times, memory data is latched and sent to the attribute controller. In graphics modes, the memory data is converted from parallel to serial bit-plane data before being sent; in alphanumeric modes, the parallel attribute data is sent.

During system accesses of video memory, the graphics controller can perform logical operations on the memory data before it reaches video memory or the system data bus. These logical operations are composed of four logical write modes and two logical read modes. The logical operators allow enhanced operations, such as a color compare in the read mode, individual bit masking during write modes, internal 32-bit writes in a single memory cycle, and writing to the display buffer on non-byte boundaries.

The *attribute controller* takes in data from video memory through the graphics controller and formats it for display. Attribute data in alphanumeric mode and serialized bit-plane data in graphics mode are converted to an 8-bit color value.Each color value is selected from an internal color palette of 64 possible colors (except in 256-color mode). The color value is used as a pointer into the video DAC where it is converted to the analog signals that drive the display.

6. Device-specific Configuration Variables

None.

7. Device Nodes

7.1. Properties

As specified in [1], [3] and [6], with the following additions or modifications.

7.1.1. Open Firmware-defined Properties for Device Nodes

`"name"` **S**

Standard *property name*, specifies the generic name of the device.

The meaning of this property is as defined in Open Firmware core document [1], as modified by the *Generic Names Recommended Practice* [5]. The value for nodes described by this specification shall be `"display"`.

`"device_type"` **S**

Standard *property name* to define the device's implemented interface.

The meaning of this property is as defined in the Open Firmware core document [1]. The value for nodes described by this specification shall be "`display`".

`"compatible"` **S**

Standard *property name*, specifies device names with which this device is compatible.

The meaning of this property is as defined in Open Firmware, as modified by the *Generic Names Recommended Practice* [5]. As described in those documents, the entries are a list of device names with which this device is compatible, starting with the name of the device itself and progressing through successively less precise and possibly less functional compatible devices.

The value of this property shall include "`pnpPNP,900`".

Additional entries may be supplied, at their appropriate position in the list, to describe devices with which this device is compatible.

If the device is on the PCI bus, the value of the compatible property shall also include, prior to "`pnpPNP,900`", an entry identifying the chip that is directly connected to the PCI bus, in the form defined by the PCI Open Firmware binding [3], and, if possible, prior to the PCI chip identifying entry, an entry identifying the overall board design. The chip identity itself is often insufficient to describe the complete device programming model, because of the presence of other programmable components such a color lookup tables and clock generators.

A device which specifies that it is compatible with this device is indicating that it will operate in a VGA compatible mode as explained in [4].

`"reg"` **S**

Standard *property name* to define the package's registers.

The meaning of this property is as defined in the Open Firmware core document [1]. It describes the device's register set. The values which shall be assigned to this property are explained in the *PCI binding*[3] and the *I/O Device Reference*[4]

7.1.2. Device-specific Properties for Device Nodes

None.

7.2. Methods

7.2.1. Open Firmware-defined Methods for Device Nodes

As specified in [1], without addition or modification.

7.2.1.1. Device Arguments for "Open" Method

As specified in [1], without addition or modification.

7.2.2. Device-specific Methods for Device Nodes

None.

8. User Interface Commands

8.1. Open Firmware-defined User Interface Commands

None.

8.2. Device-specific User Interface Commands

None.

9. Device State

9.1. Device State When Client is Started

For devices not selected as Open Firmware's "console input device" or "console output device" device, most of the initial state is typically undefined.

For devices which are selected as Open Firmware's "console input device" or "console output device" device, the device shall be initialized appropriately for the device to which it is attached.

Refer to [4] for more information on the state of this device when the client is started.

9.2. Device State Required When Client Calls Open Firmware

For devices not selected as Open Firmware's "console input device" or "console output device" device, there is no requirement.

For devices selected as Open Firmware's "console input device" or "console output device" device, the state should be unmodified from the initial state on client start-up.

Note: If the device is in a different state when the client calls Open Firmware, unpredictable behavior may result if Open Firmware accepts input or generates output. Clients changing the device state should either restore the original state before calling Open Firmware or should avoid using Open Firmware features requiring user interaction. Changing the device state is likely to render Open Firmware unusable for debugging purposes.

10. Other Commentary