PC Card Binding to IEEE Standard 1275-1994, Standard for Boot (Initialization Configuration) Firmware

Draft Revision: 1.2

Date: 03/08/95

This page intentionally blank.

Introduction

Firmware is the ROM-based software that controls a computer between the time it is turned on and the time the primary operating system takes control of the machine. Firmware's responsibilities include testing and initializing the hardware, determining the hardware configuration, loading (or booting) the operating system, and providing interactive debugging facilities in case of faulty hardware or software.

Historical Perspective

Historically, firmware designs have been proprietary and often specific to a particular bus or instruction set architecture (ISA). This need not be the case. Firmware can be designed to be machine-independent and easily portable to different hardware. There is a strong analogy with operating systems in this respect. Prior to the advent of the portable UNIX operating system in the mid-seventies, the prevailing wisdom was that operating systems must be heavily tuned to a particular computer system design and thus effectively proprietary to the vendor of that system.

IEEE Standard 1275-1994 Standard for Boot (Initialization Configuration) Firmware, Core Requirements and Practices (referred to in the remainder of this document as Open Firmware) is based on Sun Microsystems's OpenBoot firmware. The OpenBoot design effort began in 1988, when Sun was building computers based on three different processor families; thus, OpenBoot was designed from the outset to be ISA-independent (independent of the Instruction Set Architecture). The first version of OpenBoot was introduced on Sun's SPARCstation 1 computers. Based on experience with those machines, OpenBoot version 2 was developed, and was first shipped on SPARCstation 2 computers. This standard is based on OpenBoot version 2.

Purpose and Features of the Open Firmware Specification

Open Firmware has the following features:

- A mechanism for loading and executing programs (such as operating systems) from disks, tapes, network interfaces, and other devices.
- An ISA-independent method for identifying devices "plugged-in" to expansion buses, and for providing firmware and diagnostics drivers for these devices.
- An extensible and programmable command language based on the Forth programming language.
- Methods for managing user-configurable options stored in non-volatile memory.
- A "call back" interface allowing other programs to make use of Open Firmware services.
- Debugging tools for hardware, firmware, firmware drivers, and system software.

Purpose of this Bus Binding

- This document specifies the application of Open Firmware to the PC Card Bus, including PC Card-specific requirements and practices for address format, interrupts, probing, and related properties and methods.
- The core requirements and practices specified by the Open Firmware standard must be augmented by system-specific requirements to form a complete specification for the firmware for a particular system. This document establishes such additional requirements pertaining to PC Card.

Task Group Members

The following individuals were members of the Task Group that produced the first document version:

- William M. Bradley, President, FirmWorks
- Yongjae Rim, Draft Editor, IBM Corporation
- John Beidl, Task Group Leader of RIPL in PC Card Host Service Working Group, IBM Corporation
- Ron Hochsprung, Apple Computer, Inc.
- John Rible
- David L. Paktor, HaL Computer Systems, Inc.

The following individuals were added to the Task Group to produce the Document Revisions 1.1 & 1.2:

- Robert Coffin/Mike Segapeli, Draft Editors - Revision 1.1 & 1.2, IBM Corporation
- Robert Coffin, Secretary of Task Group, IBM Corporation

Trademarks

References

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

[2] PC Card Standard, January 1995 published by PCMCIA(Personal Computer Memory Card International Association)

1. Overview and References

This specification describes the application of Open Firmware to computer systems that use PC Cards conforming to *PC Card Standard, January 1995* and is to be used in conjunction with that document.

1.1 Definitions of Terms

Attribute Memory: PC Card memory area accessed when **REG#** is asserted. This space includes the CIS data and card configuration registers.

Adapter: The hardware that connects a host computer bus to 68-pin PC Card Sockets.

Bus Node: Device node that represents the interface, or "bridge," between a PC Card and its host. In general, the terms "Bus Node" and "Adapter" may be used interchangeably. "Bus node" is typically used when the emphasis is on the node as a part of the device tree, and "Adapter" is used with emphasis on the hardware implementation.

BVD: Acronym for Battery Voltage Detect

CardBus PC Card: A computer system interconnect specification that is similar to 16-bit PC Cards but has a 32-bit synchronous interface with bus mastering capabilities.

CIS: Acronym for Card Information Structure: the metaformat.

Child Node: An Open Firmware device node that represents a PC Card. Note that for multifunction 16-bit PC Cards and CardBus Cards that implement multiple functions, each individual function is realized as a node in the device tree.

Common Memory: PC Card memory area accessed when **REG#** is not asserted. This space is conventional memory space.

Host: A system which contains an Adapter with PC Card Sockets.

Interface: A set of signals used to implement a PC Card bus protocols.

LFS: Acronym for Linear File Store**.**

PC Card: A memory and/or I/O device that is compatible with the PC Card Standard. A PC Card plugs into a PC Card Socket. When a PC Card device has one function, the terms "PC Card function" and "PC Card device" can be used interchangeably.

PCMCIA: Acronym for Personal Computer Memory Card International Association.

16-bit PC Card: A card compliant with the 16-bit PC Card defined by the PC Card Standard.

REG#: A negative active input signal on pin 61 of the PC Card Socket which selects Attribute memory.

Relocatable Region: A range of the PC Card address space whose base address is established by a single base address register.

Socket: The 68-pin socket into which a PC Card is inserted.

Tuple: An elemental data block of the CIS. A tuple is composed of three fields: tuple code, tuple link and data field. The tuple code indicates the type of information contained within the data field; the tuple link is the pointer to the next tuple in the tuple chain; and the data field has the data for that tuple. The data field might not exist depending on the type of tuple.

WP: Acronym for Write-Protect

XIP: Acronym for eXecution-In-Place

Window: An area in a host computer memory or I/O space through which a PC Card may be addressed.

2. PC Card Characteristics

A PC Card is a device that plugs into a PC Card Socket. The PC Card Socket is a connector that provides a physical interface conforming to the PC Card Standard[2]. The PC Card Standard defines three types of interfaces: Memory-Only, I/O and CardBus interfaces. The Memory-Only interface defines the signals that support memory cards, but does not contain the I/O interface.

> *Note: The I/O interface was added by Release 2.0 of the PC Card Standard. PC Card's default to the Memory-Only interface to be compatible with pre-Release 2.0 hardware.*

The I/O interface requires that the Memory-Only interface also be implemented within the same Socket, and that the Memory-Only interface be selected following the initial application of Vcc to the PC Card. A host must identify the I/O device by reading the CIS and configuring the I/O device appropriately before its I/O interface becomes active. The configuration information is stored in Configuration Table Entries.

Although the CardBus protocol is significantly different from that of Memory-Only and I/O due to different functional requirements, CardBus supports the Memory-Only and I/O interfaces for backward compatibility. CardBus defines the **CCD#** (CardBus Card Detect) and **CVS#** (Card Voltage Sense) signals to detect the type of device, and establishes the proper interface.

16-bit PC Card and CardBus differ as follows:

- 16-bit PC Card defines separate data (8 or 16 bit wide) and address (26 bit wide) buses. CardBus defines a 32-bit shared bus for data and address.
- CardBus allows bus mastering.
- 16-bit PC Card defines Common Memory space, Attribute Memory space and I/O space. In addition, CardBus defines Configuration and Expansion ROM spaces.

  Note: The memory space of a CardBus card is considered a Common Memory space.

2.1 Address Space

Address spaces of a PC Card are mapped into the host address space using Windows. Typically, the Window is implemented as a set of registers in a PC Card Adapter.

16-bit PC Cards provide Common Memory, Attribute Memory and I/O spaces. The address space of an 16-bit PC Card must be mapped into host address space before it may be accessed by the host. The memory spaces are addressed with 26 bits of address to allow up to 64 Mbytes. A Common Memory space is used to allocate conventional memory. Each card may have up to 64Mbytes of Common Memory. The Common Memory space is selected when the **REG#** signal is not asserted. In addition to normal read/write operations, DMA (Direct Memory Access) is allowed to the Common Memory space.

The Attribute Memory space is selected when **REG#** is asserted. This space contains the CIS and the configuration registers of a PC Card. Access to this space is limited to a single byte on even address boundaries. With 26 bits of address, a PC Card may have up to 32 Mbytes of Attribute Memory. An implementation may use its Common Memory space as Attribute Memory space for cost reasons. For those cards, when the **REG#** signal is active the PC Card behaves as if the access is made to the Attribute Memory.

Since some processors cannot generate I/O cycles directly, I/O space accesses must be done with the register access words (e.g., **rb@**, **rw!**). It is recommended that a range of virtual addresses be set aside for use by **map-in** to I/O space devices so that the register access words can determine when an I/O cycle needs to be generated.

CardBus defines Configuration, Memory, I/O and Expansion ROM spaces. The Configuration space is 4Kbytes shared by 8 functions. Each function has 256 bytes of memory which is composed of a mandatory 64 bytes of header and 192 bytes of device dependent region. The header has the information required by the

host to configure the device, such as header type, latency, cache-line size, CIS pointer, base address registers for other spaces, etc. The CIS pointer identifies the memory space and the offset of the CIS.

Since the Configuration space often does not appear as a subset of the system's physical address space, this firmware specification provides bus-specific methods to access the Configuration space. Likewise, it provides methods for Special Cycles.

In addition to these address spaces, CardBus has another type of transaction in which the address bus is not used. Special Cycles (writes) are "broadcast" cycles in which the data conveys all of the information.

3. Open Firmware for PC Cards

Open Firmware is an Instruction Set Architecture and device independent booting mechanism. For a non-native device, Open Firmware requires FCode associated with the device to use the device for boot. The FCode is executed by the Open Firmware process to configure it as a bootable device. This section describes the process and convention for probing a PC Card to prepare the card for booting a system.

3.1 Probing Process

To be used as a boot device, a PC Card must be plugged into a PC Card Socket before the boot process starts. Although the PC Card specification supports live insertion and removal of cards, the automated Open Firmware booting process assumes the cards are neither inserted or removed from the beginning of the probing process until the end of the booting process.

The firmware must locate and execute the FCode of the PC Card Adapter to install the PC Card Adapter node in the device tree. This enables the host to turn on the power of the PC Card (using **set-socket** method). The firmware then probes the installed PC Cards to locate and install their device nodes in the device tree. 16-bit PC Cards may provide the FCode either in a special purpose Open Firmware Tuple in its CIS(CISTPL_SPCL with field TPLSPCL_ID equal to 2)or in the LFS. CardBus cards may provide FCode in Expansion ROM space.

A 16-bit PC Card may store the CIS in Attribute or Common Memory. However, the PC Card Standard requires the CIS to begin at Attribute Memory address 0. The remainder of the CIS may be located in any location of the 64M bytes of its Attribute or Common Memory spaces.

> *Note: For a PC Card Adapter node in "probe state", Open Firmware should allocate address space using the lazy algorithm in a relocatable region using **map-in** for efficient use of the host address space. Such "probe state" assignments are temporary and do not necessarily persist after the corresponding map-out.*

CardBus does not have Attribute Memory space. Instead, the CIS is located by using a pointer in the Configuration space.

3.2 Address Representations

The following address representation does not cover the issues related to a PC Card to PC Card bus bridge implementation.

3.2.1 Socket Number: 5 bits

The physical Socket number is assigned by the associated Adapter. Historically, the number of Sockets per R2 bus node is limited to 16 due to the X86 microprocessor register which is used to pass the bit-map of the assignable Socket. A CardBus Adapter, on the other hand, allows up to 32 Sockets which are analogous to the Device Numbers.

3.2.2 Function Number: 3 bits

CardBus and multifunction 16-bit PC Cards may have up to 8 logically independent functions implemented with its own independent set of configuration registers. The function number is used by multifunction 16-bit PC Cards and CardBus Cards during the configuration cycle to select the corresponding set of configuration registers to a function. Each function is uniquely identified by its function number from 0 to 7. Multifunctional 16-bit PC Cards and CardBus Cards which are selected during a configuration cycle decodes the function number field in the address to select the right set of configuration registers for that function. Multifunctional 16-bit PC Cards and CardBus Cards are required to have a function 0.

The CIS of function 0 may contain information that is global to a card. This global information in function 0 must be preceded by a function identification Tuple with function id for CardBus. For 16-bit PC Cards, Function 0 may contain information global to the card such as the manufactures id.

For a single function 16-bit PC Card, the function number is not used in addressing.

3.2.3 Register Number: 8 bits

The register number is used to access a register in the Configuration space for a CardBus card. During the configuration cycle, a card selects a register from a set of configuration registers of a selected function. The register number field is an offset in a configuration space of the function. The layout (locations and meanings of particular bits) of the device-independent configuration registers (*i.e.* the first 64-byte header) is specified by the CardBus standard. Other configuration registers are device- and implementation-specific. The standard configuration registers perform such functions as assigning Memory space and I/O space base addresses for the device's addressable regions.

> *Note: The configuration registers of a 16-bit PC Card have different roles from those of a CardBus Card. They are used to configure the hardware characteristics of the PC Card.*
>
> *Note: Since the configuration registers of a 16-bit PC Card are implemented in the Attribute Memory space, the registers are addressed using a full 26 bit Attribute Memory address. The register number is not used in addressing the configuration registers. Since the configuration registers are mapped in the Attribute Memory space, the address of a configuration register is computed as Base Address + (Register Number * 2).*

3.3 Physical Address Formats

Within this document, "address" refers to the address of an 8-bit byte.

3.3.1 Numerical Representation

The numerical representation of a PC Card address consists of two cells, encoded as follows. In the following discussion, the least-significant 32 bits of a cell are used. If the cell size is larger than 32 bits, additional high-order bits shall be zero. Bit 0 refers to the least-significant bit.

Bit number: `31` `0`

phys.hi cell: `npt00ccc 00000000 sssssfff rrrrrrrr`

phys.lo cell: `yyyyyyll llllllll llllllll llllllll`

where:

`n` 0 if the address is relocatable, 1 otherwise.

`p` 1 if the addressable region is “prefetchable”, 0 otherwise.

`t` 0 if the address is a CardBus address, or 1 if a 16-bit PC Card address.

`sssss` 5-bit Socket number.

`fff` 3-bit function number.

`ccc` the address space code, denoting which address space;

- 000 Configuration space. `n` and `p` must be 0, `t` must be 0. `rrrrrrrr` is the register number and *phys.lo* cell must be 0.
- 001 I/O Space. `p` must be 0. If `t` is 0, then `rrrrrrrr` is the region’s Base Address Register and must be 0x10, 0x14, 0x18, 0x1c, 0x20 or 0x24, and *phys.lo* cell is the 32-bit offset from the base of the relocatable region of I/O space; if `t` is 1, then `rrrrrrrr` is the Window number, `yyyyyy` is 0, and `ll..ll` is the 26-bit offset from the start address of the Window.
- 010 Memory Space. If `n` and `t` are 0, then `rrrrrrrr` is the region’s Base Address Register and must be 0x10, 0x14, 0x18, 0x1c, 0x20 or 0x24, and *phys.lo* cell is the 32-bit offset from the start of the relocatable region of 32-bit address Memory space; if `t` is 1, then `rrrrrrrr` is the Window number, `yyyyyy` is 0, and `ll..ll` is the 26-bit offset from the start of the relocatable region of 26-bit offset from the start address of the Window.
- 100 Attribute Memory. `rrrrrrrr` is the Window number, `yyyyyy` is 0, and `ll..ll` is the 26-bit offset from the start of the relocatable region of 26-bit offset from the start address of the Window.

Note: For a CardBus Card, the ‘p’ bit reflects the state of the “P” bit in the corresponding hardware Base Address register.

Note: The numerical representation specified herein contains the information needed to select a particular hardware device, specifying the format by which that information is communicated among elements of Open Firmware and client programs. A driver for a particular PC Card bus hardware implementation is free to extract that information and reformat as necessary for the hardware.

3.3.2 Text Representation

The text representation of a PC Card address is shown in Table 1.

Conversion of hexadecimal numbers from the text representation to numeric representation shall be case-insensitive, and leading zeros shall be permitted but not required.

Conversion from numeric representation to text representation shall use the lower case forms of the hexadecimal digits in the range a..f, surpressing leading zeros, and the SS form for a CardBus card shall be used for Configuration space addresses where fff is zero.

Table 1: Formats of text representation of address

format	description	Correspondence with numeric form
SS	Either an Attribute Memory space or a Configuration space of a PC Card	SS is the binary encoding of sssss. All other fields must be zero.
SS,F	Either an Attribute Memory space or a Configuration space of a PC Card's function	SS is the binary encoding of sssss. FF is the binary encoding of fff. All other fields must be zero.
SS,F,L8	An Attribute Memory space address with the numerical value for a 16-bit PC Card	t is 0. ccc is 100. SS is the binary encoding of sssss. F is the binary encoding of fff. L8 is the binary encoding of *phys.low*.
[n]iSS,F,RR,L8	An I/O space address with the numerical value for a CardBus card	ccc is 001. SS is the binary encoding of sssss. F is the binary encoding of fff. RR is the binary encoding of rrrrrrrr. L8 is the binary encoding of *phys.low*.
[n]m[p]SS,F,RR,L8	An address with the numerical value in either Common Memory space of a 16-bit PC Card or a memory space of a CardBus card	t is 0. ccc is 010. SS is the binary encoding of sssss. F is the binary encoding of fff. RR is the binary encoding of rrrrrrrr. L8 is the binary encoding of *phys.low*.

3.3.3 Unit Address Representation

The "unit-number" (i.e. the first component of its "reg" value) is the address of the Attribute or Configuration memory of a function. Since the "unit-number" is the address that appears in an Open Firmware 'device path', only the SS or SS,FF forms of the text representation appears in a 'device path'.

3.4 Definitions for an Adapter Node

3.4.1 Open Firmware-defined Properties

The following standard properties, as defined in Open Firmware, have special meanings or interpretations for PC Card.

"device_type" **S**

Standard *prop-name* to specify the implemented interface.

prop-encoded-array: a string encoded as with **encode-string**.

The meaning of this property is as defined in Open Firmware. A Standard Package conforming to this specification and corresponding to a device that implements a PC Card bus shall implement this property with the string value "**pccard**".

"reg" **S**

Standard *prop-name* to define the package's unit-address.

3.4.2 Bus-specific Properties

The following properties shall be provided for an Adapter node:

"#windows" **S**

prop-name, denotes the maximum number of Windows that the Adapter provides for each Socket.

prop-encoded-array: an integer encoded as with **encode-int**.

"#sockets" **S**

prop-name, denotes the number of Sockets that the Adapter supports.

prop-encoded-array: an integer encoded as with **encode-int**.

"release-level" **S**

prop-name, denotes the PC Card specification release level that the Adapter is compliant to.

prop-encoded-array: an integer encoded as with **encode-int**.

"status-change-int-caps" **S**

prop-name, denotes the bit-maps of events that can trigger a Status Change interrupt. The mapping between bits in the bit-map and the events are defined in Table 2. If a bit is set it indicates that the Socket is able to generate an interrupt when the corresponding event occurs.

prop-encoded-array: an array of integers encoded as with ***encode-int***.

Note: Bit numbering should conform to the corresponding PC Card Socket Services functions; except Socket Service functions that are not relevant have been omitted. In the case of a discrepancy, Socket Services nomenclature shall apply.

Note: There is an implied correspondence between the order of entries in the "status-change-int-caps" property and the order of the Socket number. This correspondence is implied on the following properties:

Table 2: Bit map definition for Status Change event

bit	name	description
0 (LSB)	**SBM_WP**	PC Card WP (Write-protect) signal.
1	**SBM_LOCKED**	Externally generated signal indicating the state of a mechanical or electrical card lock mechanism.
2	**SBM_EJECT**	Externally generated signal indicating a request to eject a PC Card from the Socket has been made.
3	**SBM_INSERT**	Externally generated signal indicating a request to insert a PC Card into the Socket has been made.
4	**SBM_BVD1**	PC Card BVD1 signal. When set, this signal indicates the battery is no longer serviceable.
5	**SBM_BVD2**	PC Card BVD2 signal. When set, this signal indicates the battery is weak.
6	**SBM_READY**	PC Card Ready signal.
7	**SBM_CD**	PC Card CD signal.

"**status-change-report-caps**" **S**

prop-name, denotes the bit-map status change events that the Socket is capable of reporting. The mapping between bits in the bit-map and the events is defined in Table 2. If a bit is set it indicates that the Socket is able to report the corresponding event.

prop-encoded-array: an array of integers, encoded as with **encode-int**.

"**cntl-ind-caps**" **S**

prop-name, denotes the bit-map of controls and indicators that the Socket is capable of. The mapping between bits in the bit-map and the controls or indicators is defined in Table 3 . If a bit is set it indicates that the Socket is able to control or indicate the corresponding event.

prop-encoded-array: An array of integers, encoded as with **encode-int**.

Table 3: Bit map definition of Control and Indicator Capability

bit	name	description
0 (LSB)	**SBM_WP**	Indicator for PC Card WP signal.
1	**SBM_LOCKED**	Indicator for an externally-generated signal indicating the state of a mechanical or electrical card lock mechanism.
2	**SBM_EJECT**	Control for a motor to eject a PC Card from the Socket.
3	**SBM_INSERT**	Control for a motor to insert a PC Card into the Socket.
4	**SBM_LOCK**	Control for card lock.
5	**SBM_BATT**	Indicator for state of BVD1 and BVD2.
6	**SBM_BUSY**	Indicator for showing card is in use.
7	**SBM_XIP**	Indicator for XIP application in progress.

"**interrupt-routing**" **S**

prop-name, denotes the bit-map of 16 interrupt levels that the PC Card interrupt can be routed to on a Socket. If a bit is set it indicates that the Socket is able to route the PC Card interrupt to the corresponding interrupt level.

prop-encoded-array: an array of pair of integers, encoded as with **`encode-int`**.

"**adapter-caps**" **S**

prop-name, denotes the bit-map that describes whether a control is at an Adapter level or a Socket level. The description of the related controls are provided in Table 4. If a bit is set it indicates that the Socket is able to control or indicate the corresponding event.

Table 4: Control Capability

bit	name	description
0 (LSB)	**AC_IND**	If AC_IND is set, indicators for SBM_WP, SBM_LOCKED, SBM_BATT, SBM_BUSY, and SBM_XIP are shared by all Sockets. If AC_IND is reset, there are individual indicators for each Socket.
1	**AC_PWR**	If AC_PWR is set, the Adapter sets the power level to the same value for all the Sockets. If AC_PWR is reset, the voltage level can be controlled individually for each Socket. Note that, the **set-socket** method will change the voltage of all sockets independent of the Socket parameter.
2	**AC_DBW**	If AC_DBW is set, all Windows on the Adapter must use the same data bus width. If AC_DBW is reset, the data bus width is set individually for each Window on the Adapter.
3	**AC_CARDBUS**	If AC_CARDBUS is set, all sockets are CardBus PC Card-capable. If reset, then all sockets are not CardBus PC Card-capable.

"**vcc-levels**" **S**

prop-name, denotes the encode form of v_{cc}, v_{pp1} and v_{pp2} connections allowed.

prop-encoded-array: array of (n $e_0..e_{n-1}$), where n is the number of entries, encoded as with **encode-int**, and e_i is the allowed vcc-levels, encoded as with **encode-int**. The e_i is computed as follows:

h# vcc a << vpp1 9 << vpp2 8 << or v or where vcc, vpp1 and vpp2 are binary bit values representing the allowed power signals on PC Card interface, and v is the voltage level allowed on that signal. The vcc value of 1 indicates that the vcc signal can be set to voltage v. For example, entry e whose value is 10100000101 in binary indicates that 5V is allowed to set on vcc and vpp2 pins but not on vpp1 pins at the same time.

3.4.3 Open Firmware-defined Methods for Adapter Nodes

A Standard Package implementing the "pccard" device type shall implement the following standard methods as defined in Open Firmware, with the physical address representations as specified in section 2.1 of this standard, and with additional PC Card specific semantics:

open (-- flag) **M**

Prepare this device for subsequent use

close (--) **M**

Close this previously-open device

map-in (phys.low phys.hi size -- virt) **M**

Map the specified subregion

map-out (virt size --) **M**

Destroy mapping from previous map-in

dma-alloc (size -- virt) **M**

Allocate a memory region for later use

dma-free (virt size --) **M**

Free memory allocated with dma-alloc

dma-map-in (virt size cacheable -- devaddr) **M**

Convert virtual address to device bus DMA address

dma-map-out (virt devaddr size --) **M**

Free DMA mapping set up with dma-map-in

dma-sync (virt devaddr size --) **M**

Synchronize (flush) DMA memory caches

probe-self (arg-str arg-len reg-str reg-len fcode-str fcode-len --) **M**

Interpret FCode, as a child of this node

decode-unit (addr len -- phys.lo phys.hi) **M**

Convert text representation of address to numerical representation

encode-unit (phys.lo phys.hi -- addr len) **M**

Convert numerical representation of address to text representation

3.4.4 Bus-specific Methods

A PC Card bus Adapter is abstracted by the Socket Services software layer. Socket Services provides basic configuration functions such as Window setting, power control, memory mapping, I/O mapping, interrupt steering etc. To boot a system from a PC Card, the same functionality must be implemented by bus adapter FCode. The methods described in this section define the functions required to configure a PC Card and boot a system from the PC Card.

The formats of the parameters shown in the method's stack representations are defined as follows;

- unit-number: A numeric-form of unit number encoded as with **encode-unit**.
- vcc-level: An encoded integer that describes the voltage levels of a Socket. The encoding scheme is described in Section 3.4.2.
- status-change-int: A bit-map that describes the status change interrupt setting for a Socket. The format of the bit-map is defined in Table 2. The value is an integer encoded as with **encode-int**.
- status-change-report: A bit-map that describes the status change report setting for a Socket. The format of the bit-map is defined in Table 2. The value is an integer encoded as with **encode-int**.
- cntl-ind: A bit-map that describes the control and indicator setting for a Socket. The format of the bit-map is defined in Table 2. The value is an integer encoded as with **encode-int**.
- interface-type: 0 if Memory interface only, or 1 if I/O and Memory interface.
- window-number: An integer that identifies a Window of the Bus Node. The value is encoded as with **encode-int**.
- window-state: A bit-map that describes the state of a Window. The format of the bit-map is defined in Table 5 The value is an integer encoded as with **encode-int**
- speed: An integer of bus speed in nano-seconds, encoded as with **encode-int**.
- window-capability: A bit-map that describes the capability of a Window. The format of the bit-map is defined in Table 6. The value is an integer encoded as with **encode-int**.
- rd-state: A bit-map that describes the instantaneous state of the Socket and PC Card, if inserted. The format of the bit-map is defined in Table 7. The value is an integer, encoded as with **encode-int.**

Table 5: Bit map definition for Window state

bit	description
0 (LSB)	If set, Window maps registers on a PC Card into the host system's I/O space. If reset, Window maps Common or Attribute Memory on a PC Card into the host system's memory space.
1	If set, Window is enabled and maps a PC Card into the host system memory or I/O space. If reset, Window is disabled.
2	If set, Window is programmed for a 16-bit data bus width. If reset, Window is programmed for an 8-bit data bus width.

Table 6: Bit-map definition for Window capability

bit	description
0 (LSB)	If set, Window may be used to map the Common Memory plane of a PC Card into the host system memory space.
1	If set, Window may be used to map the Attribute Memory plane of a PC Card into the host system memory space.
2	If set, Window may be used to map I/O ports on a PC Card into the host system I/O space.
3	If set, Windows uses the **WAIT#** signal from a PC Card to generate additional wait states.

Table 7: Bit map definition of Control and indicator Capability

bit	description
0 (LSB)	PC Card WP (Write-protect) signal.
1	Externally generated signal indicating the state of a mechanical or electrical card lock mechanism.
2	Externally generated signal indicating a request to eject a PC Card from the Socket has been made.
3	Externally generated signal indicating a request to insert a PC Card into the Socket has been made.
4	PC Card BVD1 signal. When set, this signal indicates the battery is no longer serviceable.
5	PC Card BVD2 signal. When set, this signal indicates that the battery is weak.
6	PC Card RDY/BSY signal.
7	PC Card card detect signal.

set-socket (unit-number vcc-level status-change-int status-change-report cntl-ind int-routing interface-type -- rc | 0)

To read the CIS initially, the parameters must be set as follow:

1. status-change-int shall mask out all interrupts.
2. vcc and vpp levels are based on the **VS1#** and **VS2#** signal pins.
3. status-change-report shall be reset and does not latch-in the status change event.
4. int-routing shall be disabled.
5. interface-type shall be set for Memory-only interface

Once the CIS is read, the card may be configured appropriately.

get-socket (unit-number -- vcc-level status-change-int status-change-report cntl-ind int-routing interface-type rc|0)

set-window (window-number window-state speed -- rc | 0)

To read the CIS initially, the parameters must be set as follows:

1. Bit 0 in window-state must be reset.
2. The speed of the Window must be as appropriate for the card type installed (300 ns. for 5 volt 16-bit PC Card).
   "set-window" does not establish the address mapping while the "set-window" defined in Socket service is establishing address mapping. During boot, map-in and map-out are used to establish the address mapping.

get-window (window-number --rc | window-state speed 0)

inquire-window (window-number addr -- rc | window-capability unit-number 0)

reset-socket (unit-number - rc | 0)

get-status (unit-number -- rc | card-state status-change-report cntl-ind int-routing interface-type 0)

special-! (data unit-number --)

Performs a CardBus Special Cycle on the indicated bus.

3.5 Definitions for a Child Node

3.5.1 Open Firmware-defined Properties

The following properties, as defined in Open Firmware, have special meanings or interpretations for PC Card.

"name" **S**

Standard *prop-name* to specify the implemented interface.

prop-encoded-array: construct a concatenated string encoded as with **encode-string** of the form

pccardVVVV,DDDD where VVVV is the manufactures id field and DDDD is the manufactures information field as defined below:

- VVVV string is defined by the Field 'TPLMID_MANF' in Tuple 'CISTPL_MANFID'.
- DDDD string as defined by the Field 'TPLMID_CARD' in Tuple 'CISTPL_MANFID'.

The VVVV and DDDD strings are ASCII hexadecimal, lower case, and without leading zeros.

Note: If the PC Card is bootable, the Open Firmware Working Group strongly recommends that Tuple 'CISTPL_MANFID' be implemented. If the PC Card doesn't implement the Tuple 'CISTPL_MANFID', then Open Firmware will create and/or use the string 'unknown'.

"reg" **S**

Standard *prop-name*, defines device's addressable regions.

prop-encoded-array: an arbitrary number of (*phys-addr size*) pairs.

phys-addr is (*phys.lo phys.hi*), encoded as with **encode-phys**. *size* is an integer, encoded as with **encode-int**.

The first entry of the property value shall be the unit-number of a function decoded as with **decode-unit**. Additional entries shall specify the address space of other addressable regions.

If a CardBus card function has an addressable region that can be accessed relative to more than one base address (for example, in Memory space relative to one Base Register, and in I/O space relative to another), only the primary access path (typically, the one in Memory space) shall be listed in the **"reg"** property, and the secondary access path shall be listed in the **"alternate-reg"** property.

"interrupts" **S**

prop-name, the presence of which indicates that the device represented by this node is connected to an interrupt line.

prop-encoded-array: integer, encoded as with **encode-int**.

3.5.2 Bus-specific Properties

Standard Packages corresponding to devices that are children of a PC Card bus shall implement the following properties, if applicable.

"16bitcard" **S**

prop-name, exists if the node implements the 16-bit PC Card interface.

prop-encoded-array: <none>

"cardbus" **S**

prop-name, exists if the node implements CardBus.

prop-encoded-array: <none>

"alternate-reg" **S**

prop-name, defines alternate access paths for addressable regions.

prop-encoded-array: an arbitrary number of (*phys-addr size*) pairs.

phys-addr is (*phys.lo phys.hi*), encoded as with **encode-phys.** *size* is an integer, encoded as with **encode-int**.

This property describes alternative access paths for the addressable regions described by the **"reg"** property. Typically, an alternative access path exists when a particular part of a device can be accessed either in Memory space or in I/O space, with a separate base address for each of the two access paths. The primary access paths are described by the **"reg"** property, and the secondary access paths, if any, are described by the **"alternate-reg"** property.

If the number of **"alternate-reg"** entries exceeds the number of **"reg"** property entries, the additional entries denote addressable regions that are not represented by **"reg"** property entries, and are thus not intended to be used in normal operation. Such regions might, for example, be used for diagnostic functions. If the number of **"alternate-reg"** entries is less than the number of **"reg"** entries, the regions described by the extra **"reg"** entries do not have alternative access paths. An **"alternate-reg"** entry whose *phys.hi* component is zero indicates that the corresponding region does not have an alternative access path. Such an entry can be used as a "place holder" to preserve the positions of later entries relative to the corresponding **"reg"** entries. The first **"alternate-reg"** entry, corresponding to the **"reg"** entry describing the function's Configuration space registers, shall have a *phys.hi* component of zero.

"windows" **S**

Standard *prop-name*, defines device's addressable regions.

prop-encoded-array: an arbitrary number of (*window base size*).

window is an integer, encoded as with **encode-int**, that denotes the Window that is assigned for the Socket which the card is plugged in. *base* is an integer encoded as with **encode-int**, that denotes the base address of the parent where the Window is mapped. *size* is an integer, encoded as with **encode-int**, that denotes the size of the address space within the Window.

Entries of the property value describe the parent's address spaces that are currently mapped to the address spaces of a PC card.

A Window shall be allocated for active address spaces only to avoid the waste of the system address space. The active spaces in this context are the address spaces that are used by functions relevant to system boot. Due to the limited number of Windows in most implementations, the Socket service allows the address space mapped in a Window be shared by multiple functions of a multifunction PC Card. If the address spaces which share a Window are not contiguous the system address space is often wasted.

If an implementation of FCode device driver requires more Windows than available, it is the responsibility of the FCode driver to manage Window allocation. The “reg” property shall describe all the address spaces used by the PC Card to avoid conflict on address space allocation for other devices.

3.5.2.1 CardBus-type Node Properties

The following properties are created for a CardBus card node during the probing process, after the device node has been created but before evaluating the device's FCode (if any). They represent the values of standard PC Card configuration registers.

“devsel-speed” **S**

prop-name, denotes the timing of **CDEVSEL#** signal.

prop-encoded-array: an integer, encoded as with **encode-int**. 0, 1, or 2 denotes the fast, medium and slow signal timing, respectively.

“fast-back-to-back” **S**

prop-encoded-array: <none>

This property shall be present if the “fast-back-to-back” bit (bit 7) in the function's Status Register is set, and shall be absent otherwise.

3.6 Bus-specific User Interface Commands

An Open Firmware-compliant User Interface on a system with built-in PC Card *should* implement the following PC Card-specific user interface commands.

probe-pccard-socket (unit-number --)

Enter “probe state”, thus affecting subsequent behavior of the **map-in** and **map-out** methods.

For a card plugged in the Socket, the CIS of the card is searched and the bus-specific node properties are created if appropriate. If the card has an FCode program, the FCode image for the card is constructed in the RAM and is consistent as defined in the probing process, Section 3.1.

For a CardBus card, if a FCode program is not found, the process shall determine whether or not the function has an expansion ROM image containing a FCode program. If the function has an appropriate FCode program, the FCode program from expansion ROM is copied into the RAM.

When the FCode program starts execution, **my-address** and **my-space** shall return the address of the Attribute Memory space or the Configuration space of the function.
The FCode in the RAM is evaluated as with **byte-load**. The FCode program shall create the **“name”** and **“reg”** properties.
If the function does not have a FCode program, the firmware shall create the value of the **“name”** property. For this case, the name shall be created as defined in Open Firmware-defined Properties, Section 3.5.1.

The first entry of the **“reg”** property value shall be the unit-number of the function. Additional entries shall specify the address space of other addressable regions.

Without FCode, it is not necessarily possible to determine whether or not there are multiple base address registers mapping the same resource, so it is not possible to create an **“alternate-reg”** property.

The configuration register is set if necessary. Assign address spaces for the corresponding Child Node.

probe-pccard (--)
Scan all Sockets in numerical order as with **probe-pccard-socket**.