PC CARD STANDARD

Volume 8

PC Card Host System Specification

REVISION HISTORY

Date	PC Card Host System Specification Version	PC Card Standard Release	Revisions
03/97	6.0	6.0 Release	Initial version Added PC Card Host Thermal Ratings
04/98	6.1	6.1 Update	PCI Bus Power Management for CardBus Bridges PCI-to-CardBus Bridge Register Description
02/99	7.0	7.0 Release	Corrections to PCI Bus Power Management for CardBus Bridges
03/00	7.1	7.1 Update	Modified Peak, Static, and Average Current Definitions Redefined Peak, Static, and Average Ipp Current Values
11/00	7.2	7.2 Update	Added Standardized Zoomed Video Register Model Added PC Card Socket Physical Specification
04/01	8.0	8.0 Release	Added **VCORE** Supplemental Voltage Capability Added 1.8 V option for **VCORE** Changed to PC Card Standard Volume Number 8 (was Volume Number 11)

©2001 PCMCIA / JEITA

All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording or otherwise, without prior written permission of PCMCIA and JEITA. Published in the United States of America.

CONTENTS

©2001 PCMCIA/JEITA

©2001 PCMCIA/JEITA

©2001 PCMCIA/JEITA

FIGURES

©2001 PCMCIA/JEITA

TABLES

©2001 PCMCIA/JEITA

©2001 PCMCIA/JEITA

1. INTRODUCTION

1.1 Purpose

The PC Card Host System Specification intends to define requirements for any host system containing a PC Card socket. This currently includes the following applications:

- PC Card Host Thermal Ratings
- PCI Bus Power Management Interface for PCI-to-CardBus Bridges
- PCI-to-CardBus Bridge Registers
- PC Card Socket Physical Specification

1.2 Related Documents

The following documents which comprise the ***PC Card Standard***:

PC Card Standard Release 8.0 (April 2001), PCMCIA /JEITA
Volume 1. ***Overview and Glossary***
Volume 2. ***Electrical Specification***
Volume 3. ***Physical Specification***
Volume 4. ***Metaformat Specification***
Volume 5. ***Card Services Specification***
Volume 6. ***Socket Services Specification***
Volume 7. ***PC Card ATA Specification***
Volume 8. ***PC Card Host Systems Specification***
Volume 9. ***Guidelines***
Volume 10. ***Media Storage Formats Specification***
Volume 11. ***XIP Specification***

PCI Local Bus Specification, Revision 2.1, June 1, 1995, PCI Special Interest Group

PCI to PCI Bridge Architecture Specification, Revision 1.0, April 5, 1994, PCI Special Interest Group

PCI Mobile Design Guide, Revision 1.0, October 27, 1994, PCI Special Interest Group

PCI Bus Power Management Specification, Revision 1.0, Mar 18, 1997, PCI Special Interest Group

Advanced Configuration and Power Interface Specification Revision 1.0, Intel Corporation, Microsoft Corporation and Toshiba

Toward the "OnNow" Machine: The Evolution of the PC Platform, Microsoft Technology Brief, April 1996, Microsoft Corporation

Device Power Management, Microsoft Technology Brief, April 1996, Microsoft Corporation

Device Class Power Management Reference Specifications, Draft Proposals, 1996, Microsoft Corporation

OnNow Design Initiative and ACPI, Microsoft Web Page with links to many of the documents above, 1996, Microsoft Corporation

OnNow Power Management and the Win32 Driver Model, 1996, Microsoft Corporation

©2001 PCMCIA/JEITA

PCI Hot-Plug Specification, PCI Special Interest Group

ExCA Compliance Test Procedure and Specification

©2001 PCMCIA/JEITA

2. SYSTEM THERMAL AND POWER

This section defines a low cost optional method which can be used in determining the host platform thermal rating. The purpose of determining the thermal rating is to ensure that the heat generated and dissipated within the body of the PC Card does not thermally exceed the capabilities of the host system to remove excessive heat in order to maintain the PC Card at an acceptable temperature limit. The host system method of choice shall be run at the manufacturer's worst case environmental upper limits for the product being rated.

2.1 Host System Thermal Measurement

The purpose of determining the host system's thermal rating is to measure the ability of the host to remove heat from the PC Card environment and maintain the PC Card surface temperature. This measurement is performed using a thermal transmission device of the same size and physical dimensions of a PC Card with an external power source supplying the energy to raise the transmission device's surface temperature.

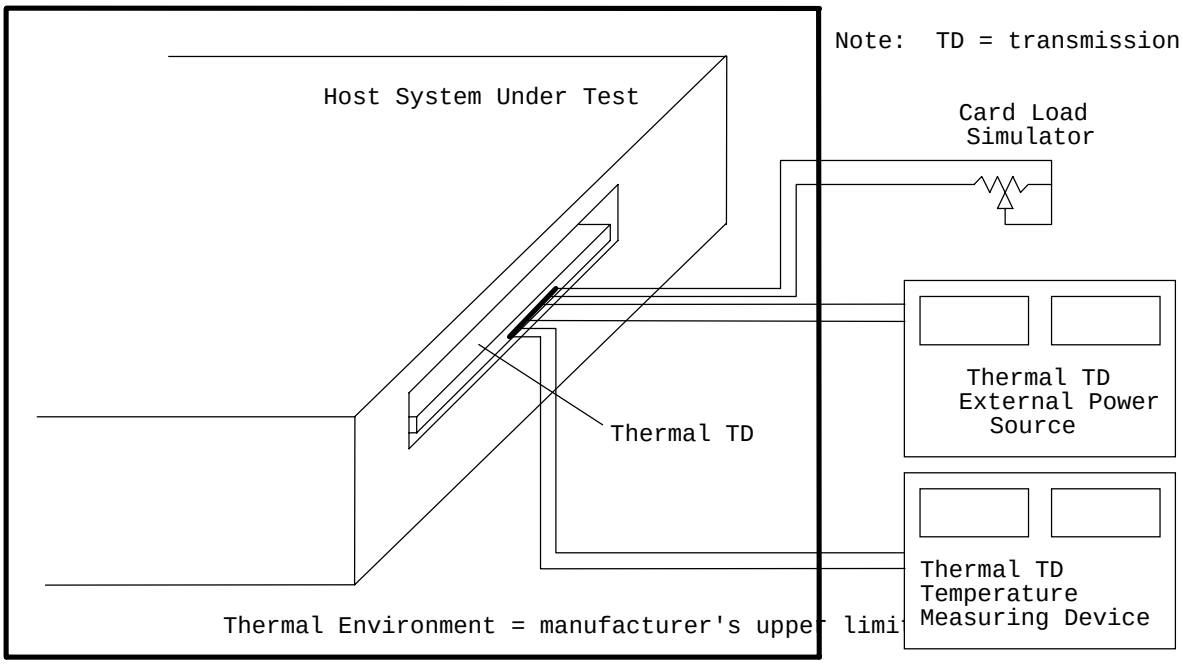


Figure 2-1: Determing System Thermal Rating: Construction of the Host System Thermal Rating Environment

A thermal transmission device must be used to introduce heat into the host system in order to determine the host system's ability to remove a PC Card's heat energy. A thermal input device in the form factor of a Type II 16-bit (not CardBus) PC Card shall be used. The thermal input device can be made from a Type II card frame and metal side panels using industry standard frames and panel kits. The thermal input device shall be connected to all PC Card pins except the **VCC**, **VPP/VCORE** and card detect pins. A Kapton™ heater element or similar device is used as the heater element for the thermal

input device. An adjustable external power supply is used to drive the heater and an external temperature measuring device is used to monitor the thermal input device's surface temperature. Lastly, a device must be utilized to draw as much power from the host system power supply as what is being input to the thermal input device (see ***Figure 2-1***). This is a simple variable load that may be connected to a second slot **VCC/VPP/VCORE** pins through a dummy card if available.

2.1.1 Procedure

Power on the host system unit and run a program, designed to prevent the host platform from activating any power management features that maintains activity. The host system must be placed on a thermally non-conductive surface.

Introduce external heat into the PC Card environment via the external power supply. Adjust the PC Card Load Simulator to consume as much power as that being introduced into the thermal input device. When the temperature of the thermal input device stabilizes at 65°C ± 1°C for a sixty minute period, the thermal rating of the system has been reached. The DC power reading is the thermal rating (voltage X current) which is recorded and reported in the Host System Data Table (see Section ***2.2 Host System Capabilities and the Host System Data Table***).

©2001 PCMCIA/JEITA

2.2 Host System Capabilities and the Host System Data Table

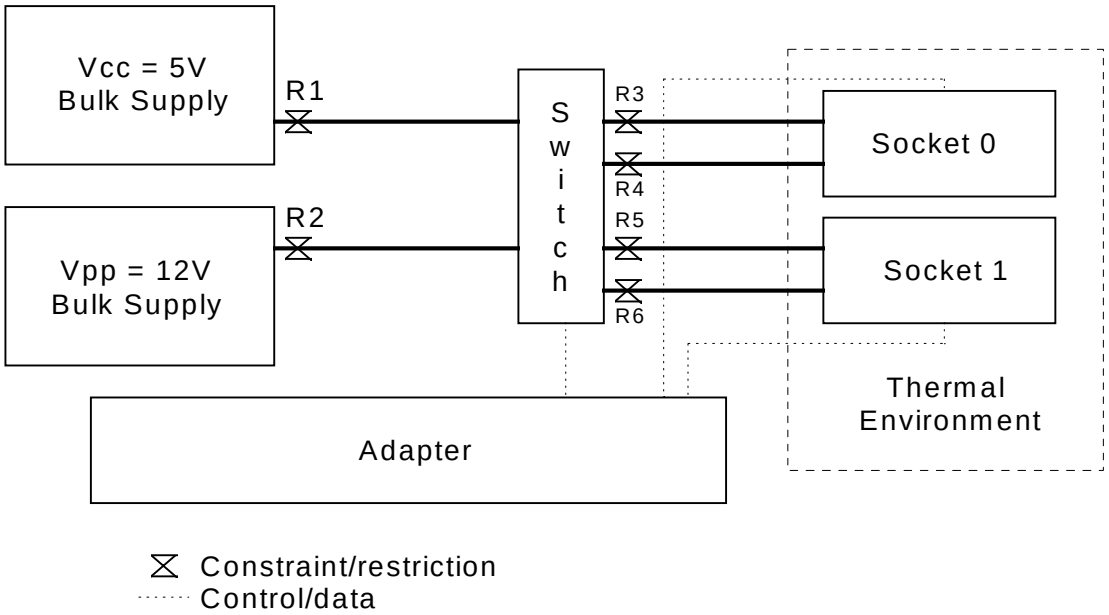


Figure 2-2: Typical Environment

In a typical environment, power sources are limited by the design's allotment of bulk power supply current to the PC Card sub-system and the current capability of the switching mechanism turning on **VCC** and **VPP/VCORE** to the PC Card. Restrictions R1 and R2 represent the dedicated PC Card design allotment and restrictions R3 and R4 represent the limitations of the switching circuit. When distributing power resources, the controlling software must ensure that neither the switch limit nor the resource supply allocation limits are violated. It is the host's responsibility to supply the design limits to the PC Card controlling software.

A thermal environment is limited is imposed also by design. This limit is to ensure that neither the PC Card nor the host system is damaged by excessive heat.

The thermal rating indicates the ability of the host system to remove heat. This rating represents the maximum amount of heat the unit can remove and maintain the card at a maximum surface temperature of 65° C at the manufacturer's environmental upper limit. The format of the thermal rating values is defined using the mantissa - exponent - extension (see the ***Metaformat Specification***).

The power capabilities described within this data structure represent not only what voltages are available, but also the amount of current the host is capable of providing, and what, if any, limitations there may be due to voltage switches, connector contacts, etc.

The data structure describing the host capabilities shall be located in the Host System Data Table.

©2001 PCMCIA/JEITA

2.2.1 Data Record Header

Table 2-1: Data Record Header

Host System PCMCIA Capabilities	
Record ID	
Ext	Thermal and Power Record Number (6::0)
Version	
major	minor
8-Bit Checksum	
Checksum	
Byte count of total data structure (including ID)	
Byte count (LSB)	
Byte count (MSB)	

The header begins with a record number. Following the record number is a record version number defined as major (0 - 15).minor (1 - 9), followed by an 8-bit checksum which results in a zero checksum. Completing the header is a 16-bit byte count of the entire data record. Version numbers and comments are as follows:

Version History

Major	Minor	Comments
1	0	Initial Release
Reserved	Reserved	Future Release Numbers

The record number is of the bit 7 extension type. Bit 7 of each byte is the extension bit (Ext) which indicates that another byte of conditions follows the present byte. Bits 0 - 6 represent ascending ordered record number bits, as required, while bit 7 in each byte indicates whether or not the next byte is an extension of the record number. The following byte represents record number xxxxxxx, no extension:

offset n + 0	0	x	x	x	x	x	x	x

while the following two bytes represent record number yyyyyyyxxxxxxx:

offset n + 0	1	x	x	x	x	x	x	x
offset n + 1	0	y	y	y	y	y	y	y

The first record would have the format:

offset n + 0	0	0	0	0	0	0	0	1

©2001 PCMCIA/JEITA

2.2.2 Base Address/Controller ID of the Controllers

Table 2-2: Base Address/Controller ID of the Controllers

Base Address/Controller ID of the controllers (as required)				
Ext	RFU(0)	Skts Enum	Bus Type (4::3)	Controller ID (2::0)
Legacy controller base address 7::0 (LSB) or Device number (4::0)				
Legacy controller base address 15::8 (MSB) or Bus number (7::0)				
Socket Enumeration				
Socket Count				
PCI Sockets				
Socket number			Device number (if required) (4::0)	
Bus number (if required) (7::0)				

The first entry of the Base Address / Controller ID is used to identify the bus type and the controller ID. Also indicated in this byte is whether or not the controller's sockets will be enumerated with location information particular to this implementation.

If bit 7 is set, more than one controller is being defined. Bit 6 is reserved for future use (0).

Bits 4::3 specify the type of system bus and are defined as follows:

Bits 4::3	Bus Type
0	ISA
1	PCI
2	RFU
3	RFU

Bits 2:0 specify the identification number this controller will use within this record and is unique to this record. There is no conformity between this record's ID and the host system software's ID or any other record's ID for this controller.

While sockets are not required to be enumerated, it is required to specify the number of sockets associated with this controller. If sockets are enumerated (such as for a PCI environment) then the Skt Enum bit, bit 5, shall be set. The socket enumeration for each bus architecture may be different and will be defined when the bus is defined.

The base address of the controllers is a physical address and is bus dependent. If the controller is a PCI agent, the bus type will be 01h and the bridge can be located by the device number and bus number used to isolate the controller. If the PCI agent supports Card Bus cards and legacy cards, the base address of the legacy controller function is determined by reading bytes 44:47 in the bridge configuration space. If the controller is an ISA only device, the bytes are used to denote the ISA beginning address and the bus type will be zero. This physical address is the mechanism which PC Card software uses in correlating the software's logical controller / socket number with the controller ID. Socket enumeration shall begin with zero so as to maintain compliance with the PCI bridge multifunction specification. Three bits of socket ID are used to define up to eight controllers. This ID is used to designate controllers within this data structure only and have no correlation with the PC Card software's or any other record's socket number. Bus types 02h and 03h are reserved for future definition.

©2001 PCMCIA/JEITA

2.2.3 Thermal Description

Every socket is contained in a thermal environment. The thermal environment indicates how much heat can be generated and still maintain the PC Card surface temperature at 65° C. The extension bit is used to indicate the presence of an extension byte (in the mantissa / exponent byte) or additional extensions.

Table 2-3: Thermal Description

Thermal Descriptions		
Ext	Thermal rating mantissa	Thermal rating exponent
Ext	Extension	
Adapter/sockets within this environment (as required)		
Ext	Controller ID (6::4)	Socket ID (3::0)

2.2.3.1 Thermal Ratings

This value is the thermal rating of the environment. This value represents the maximum amount of heat which the host system can receive from the card and still maintain a card surface temperature of 65°C or below.

2.2.3.2 Adapter/Sockets Within This Environment

Following the environment's thermal rating is a list of sockets which exist within that thermal environment. Bit 7 is used to extend the list for as many entries as is necessary. Bits 6::4 contain the controller ID as defined in the record base address / controller ID section of the data structure. Bits 3::0 define a socket of the controller which exist within this environment.

©2001 PCMCIA/JEITA

2.2.4 Vcc and Vpp Capabilities

This data structure defines the **VCC** and **VPP/VCORE** capabilities of the host system controller.

Table 2-4: Power Sources Descriptions

Power Sources Descriptions							
Voltage / Current Descriptions							
Ext	RFU(0)	RFU(0)	Static I	Ave I	Peak I	Min Max V	Nom V
Voltage Connection Options Definition							
RFU (0)	RFU (0)	RFU (0)	RFU (0)	Vcore	Vpp = Vcc	Vpp	Vcc
Voltage Description							
Ext	Nominal voltage mantissa				Nominal voltage exponent		
Ext	Nominal voltage extension						
··· or ···							
Ext	Minimum voltage mantissa				Minimum voltage exponent		
Ext	Minimum voltage extension						
Ext	Maximum voltage mantissa				Maximum voltage exponent		
Ext	Maximum voltage extension						
Current Description							
Ext	peak current mantissa				peak current exponent		
Ext	Extension						
Ext	average current mantissa				average current exponent		
Ext	Extension						
Ext	static current mantissa				static current exponent		
Ext	Extension						
User(s) of this power source (as required)							
Ext	Controller ID (6::4)				Socket ID (3::0)		
Ext	peak current mantissa				peak current exponent		
Ext	Extension						
Ext	average current mantissa				average current exponent		
Ext	Extension						
Ext	static current mantissa				static current exponent		
Ext	Extension						
Secondary power source count							
Count of power source(s) derived from this primary source							
Secondary power sources (as required)							
Voltage / Current Descriptions							
Ext	RFU(0)	RFU(0)	Static I	Ave I	Peak I	Min Max V	Nom V
Voltage Connection Options Definition							
RFU (0)	RFU (0)	RFU (0)	RFU (0)	Vcore	Vpp = Vcc	Vpp	Vcc
Voltage Description							
Ext	Nominal voltage mantissa				Nominal voltage exponent		
Ext	Nominal voltage extension						
... or ...							

Ext	Minimum voltage mantissa	Minimum voltage exponent
Ext	Minimum voltage extension	
Ext	Maximum voltage mantissa	Maximum voltage exponent
Ext	Maximum voltage extension	

Current Description

Ext	peak current mantissa	peak current exponent
Ext	Extension	
Ext	average current mantissa	average current exponent
Ext	Extension	
Ext	static current mantissa	static current exponent
Ext	Extension	
Ext	multiplier mantissa	multiplier exponent
Ext	multiplier extension	

User(s) of this power source (as required)

Ext	Controller ID (6::4)	Socket ID (3::0)
Ext	peak current mantissa	peak current exponent
Ext	Extension	
Ext	average current mantissa	average current exponent
Ext	Extension	
Ext	static current mantissa	static current exponent
Ext	Extension	

Secondary power source count

Count of power source(s) derived from this primary source

2.2.4.1 Power Sources Descriptions

Power sources are divided into two types: primary, or bulk supplies, and secondary. Secondary power sources are derived from a primary supply using such circuits as charge pumps. Secondary supplies increase the current required from the parent, thus this value must be added not only to the secondary power source accumulator, but must also be multiplied by an inefficiency number and added to the current accumulator of the parent.

©2001 PCMCIA/JEITA

2.2.4.1.1 Voltage / Current Descriptions

This byte specifies what type of voltage and whether or not current values will be defined.

Ext	RFU(0)	RFU(0)	Static I	Ave I	Peak I	Min Max V	Nom V

Bit	Description
Ext	This bit is used to indicate that more voltage definitions will follow. If the value is 1, then the definition block will be repeated for at least one more voltage definition.
RFU(0)	Reserved for future use, set to 0
RFU(0)	Reserved for future use, set to 0
Static I	This bit, if set, indicates that the static current will be specified for the voltage being defined.
Ave I	This bit, if set, indicates that the average current will be specified for the voltage being defined.
Peak I	This bit, if set, indicates that the peak current will be specified for the voltage being defined.
Min MaxV	This bit is used to denote a variable voltage is being defined, and that the maximum and minimum values will be defined. Min MaxV and Nom V are mutually exclusive.
NomV	This bit, if set, indicates the voltage being defined is not variable, and the nominal value will be defined. Min MaxV and Nom V are mutually exclusive.

2.2.4.1.2 Voltage Connection Options Definition

This byte specifies what type of voltage is being defined.

RFU (0)	RFU (0)	RFU (0)	RFU (0)	Vcore	Vpp = Vcc	Vpp	Vcc

Bit	Description
RFU(0)	Reserved for future use, set to 0
RFU(0)	Reserved for future use, set to 0
RFU(0)	Reserved for future use, set to 0
RFU(0)	Reserved for future use, set to 0
Vcore	If set (1), the voltage being defined can be connected to the **VCORE** pins of the socket. If reset (0), the voltage being defined cannot be provided to the **VCORE** pins.
Vpp1,2 = Vcc	If set (1), the voltage being defined can only be connected to the **VPP** pins of the socket when the voltage is also being used as **VCC**. If reset (0), the voltage is available to the **VPP** pins regardless of the **VCC** setting. This bit is only valid when both the **VPP** and **VCC** bits are set.
Vpp	If set (1), the voltage being defined can be connected to the **VPP** pins of the socket. If reset (0), the voltage being defined cannot be provided to the **VPP** pins.
Vcc	If set (1), the voltage being defined can be connected to the **VCC** pins of the socket. . If reset (0), the voltage being defined cannot be provided to the **VCC** pins.

2.2.4.1.3 Voltage Description

The following bytes define either the nominal voltage value or the minimum and maximum voltages. For a nominal voltage as specified by bit 0, Nom V, of the byte Voltage / Current Descriptions, the following format is used:

Ext	Nominal voltage mantissa	Nominal voltage exponent
Ext	Nominal voltage extension	

The extension bit (Ext) in the mantissa/exponent byte, if set, indicates an extension byte follows. If the extension bit (Ext) in the voltage extension byte is set, additional voltage extensions follow.

A Nominal voltage mantissa value of 07_H (111b) and a Nominal voltage exponent value of $0F_H$ (1111b) and a Nominal voltage extension value of $07F_H$ (1111111b) and no extension (bit 7 of the Nominal voltage extension = 0) indicates the voltage has a value of infinite impedance (open) with no current capability and none need be defined.

A Nominal voltage mantissa value of 0_H (000b) and a Nominal voltage exponent value of $0F_H$ (1111b) and a Nominal voltage extension value of 00_H (0000000b) and no extension (bit 7 of the Nominal voltage extension = 0) indicate the voltage has a value of 0 (zero) impedance (ground) with no current capability and none need be defined.

These values are intended to be used for specifying Vpp capabilities.

For a variable voltage, as indicated by bit 1, Min Max V, of the byte Voltage / Current Descriptions, the format to be used is:

Ext	Minimum voltage mantissa	Minimum voltage exponent
Ext	Minimum voltage extension	
Ext	Maximum voltage mantissa	Maximum voltage exponent
Ext	Maximum voltage extension	

2.2.4.1.4 Current Description

Following the voltage definition are the current definitions as indicated by bits 2 - Peak Current, 3 - Average Current and 4 - Static Current of the Voltage / Current Descriptions byte. The format is:

Ext	peak current mantissa	peak current exponent
Ext	Extension	
Ext	average current mantissa	average current exponent
Ext	Extension	
Ext	static current mantissa	static current exponent
Ext	Extension	

The values specified represent the maximum source amount for each category available which is presented to the voltage's user community.

2.2.4.1.5 User(s) of This Power Source (As Required)

The users of the power source previously defined are listed by controller and socket number. See section ***2.2.2 Base Address/Controller ID of the Controllers*** to determine the socket numbering and address scheme. Socket users are defined in terms of the socket number and the controller number. Socket specification includes peak, average and static current which is the maximum of each category which can be passed through to the socket. When set, the extension bit, bit 7 of the socket ID byte, indicates that another user socket will be defined following this definition.

©2001 PCMCIA/JEITA

Ext	Controller ID (6::4)	Socket ID (3::0)
Ext	peak current mantissa	peak current exponent
Ext	Extension	
Ext	average current mantissa	average current exponent
Ext	Extension	
Ext	static current mantissa	static current exponent
Ext	Extension	

2.2.4.1.6 Secondary Power Source Count

Secondary power sources are specified at the end of the parent primary power source specification. After all user sockets of the parent have been specified, a *Secondary power source count* is provided to indicate whether or not any secondary power sources are derived from this primary. If the *Secondary power source count* is non-zero, then the data structure following is a secondary power source. The format of the secondary power source is the same as a primary except for the addition of a multiplier. Any utilization of the secondary power source must be reflected back to the parent primary after the multiplier effect has been applied.

2.2.4.1.7 Peak, Average and Static Current Definition

See the ***Overview and Glossary*** for definition of Peak, Average, and Static Current.

Table 2-5: Recommended Minimum Current Per Slot Values[1]

Voltage	Current Type	Name	3.3V Value	5V Value
V_{CC}	Peak	I_{CCP}	1000 mA	660 mA
V_{CC}	Average	I_{CCA}	750 mA	500 mA
V_{CC}	Static	I_{CCS}	500 mA	330 mA
Voltage	**Current Type**	**Name**	**1.8V Value**	**3.3V Value**
V_{CORE}	Peak	I_{COREP}	500mA	300mA
V_{CORE}	Average	I_{COREA}	375mA	250mA
V_{CORE}	Static	I_{CORES}	250mA	200mA
Voltage	**Current Type**	**Name**	**Value for all Vpp Voltages**[2]	
V_{PP}	Peak	I_{PPP}	50mA	
V_{PP}	Average	I_{PPA}	50 mA	
V_{PP}	Static	I_{PPS}	50mA	

1. PC Cards requiring more current than the host minimum recommended support values may not be powered properly in all systems
2. Typical Vpp values are Vcc and 12V.

2.2.5 Host System Capabilities Data Structure

Table 2-6: Host System Capabilites Data Structure

Host System PCMCIA Capabilities

Record ID							
Ext	Thermal and Power Record Number (6::0)						
Version							
major				minor			
8-Bit Checksum							
Checksum							
Byte count of <u>total</u> data structure (including ID)							
Byte count (LSB)							
Byte count (MSB)							
Base Address/Controller ID of the controllers (as required)							
Ext	RFU(0)	Skts Enum	Bus Type (4::3)		Controller ID (2::0)		
Legacy controller base address 7::0 (LSB) or Device number (4::0)							
Legacy controller base address 15::8 (MSB) or Bus number (7::0)							
Socket Enumeration							
Socket Count							
PCI Sockets							
Socket number if required (7::5)			Device number (if required) (4::0)				
Bus number (if required) (7::0)							
Thermal Descriptions							
Ext	Thermal rating mantissa			Thermal rating exponent			
Ext	Extension						
Adapter/sockets within this environment (as required)							
Ext	Controller ID (6::4)			Socket ID (3::0)			
Power Sources Descriptions							
Voltage / Current Descriptions							
Ext	RFU(0)	RFU(0)	Static I	Ave I	Peak I	Min Max V	Nom V
Voltage Connection Options Definition							
RFU (0)	RFU (0)	RFU (0)	RFU (0)	Vcore	Vpp = Vcc	Vpp	Vcc
Voltage Description							
Ext	Nominal voltage mantissa (bit 0 = 1)			Nominal voltage exponent (bit 0 = 1)			
Ext	Nominal voltage extension (bit 0 = 1)						
··· or ···							
Ext	Minimum voltage mantissa (bit 1 = 1)			Minimum voltage exponent (bit 1 = 1)			
Ext	Minimum voltage extension (bit 1 = 1)						
Ext	Maximum voltage mantissa (bit 1 = 1)			Maximum voltage exponent (bit 1 = 1)			
Ext	Maximum voltage extension (bit 1 = 1)						

©2001 PCMCIA/JEITA

Current Description

Ext	peak current mantissa	peak current exponent
Ext	Extension	
Ext	average current mantissa	average current exponent
Ext	Extension	
Ext	static current mantissa	static current exponent
Ext	Extension	

User(s) of this power source (as required)

Ext	Controller ID (6::4)	Socket ID (3::0)
Ext	peak current mantissa	peak current exponent
Ext	Extension	
Ext	average current mantissa	average current exponent
Ext	Extension	
Ext	static current mantissa	static current exponent
Ext	Extension	

Secondary power source count

Count of power source(s) derived from this primary source

Secondary power sources (as required)

Voltage / Current Descriptions

Ext	RFU(0)	RFU(0)	Static I	Ave I	Peak I	Min Max V	Nom V

Voltage Connection Options Definition

RFU (0)	RFU (0)	RFU (0)	RFU (0)	RFU (0)	Vpp1,2 = Vcc	Vpp	Vcc

Voltage Description

Ext	Nominal voltage mantissa (bit 0 = 1)	Nominal voltage exponent (bit 0 = 1)
Ext	Nominal voltage extension (bit 0 = 1)	

··· or ···

Ext	Minimum voltage mantissa (bit 1 = 1)	Minimum voltage exponent (bit 1 = 1)
Ext	Minimum voltage extension (bit 1 = 1)	
Ext	Maximum voltage mantissa (bit 1 = 1)	Maximum voltage exponent (bit 1 = 1)
Ext	Maximum voltage extension (bit 1 = 1)	

Current Description

Ext	peak current mantissa	peak current exponent
Ext	Extension	
Ext	average current mantissa	average current exponent
Ext	Extension	
Ext	static current mantissa	static current exponent
Ext	Extension	
Ext	multiplier mantissa	multiplier exponent
Ext	multiplier extension	

©2001 PCMCIA/JEITA

User(s) of this power source (as required)

Ext	Controller ID (6::4)	Socket ID (3::0)
Ext	peak current mantissa	peak current exponent
Ext	Extension	
Ext	average current mantissa	average current exponent
Ext	Extension	
Ext	static current mantissa	static current exponent
Ext	Extension	

Secondary power source count

Count of power source(s) derived from this primary source

©2001 PCMCIA/JEITA

3. PCI BUS POWER MANAGEMENT INTERFACE SPECIFICATION FOR PCI-TO-CARDBUS BRIDGES

3.1 Introduction

Since its introduction in 1993, PCI has become a very popular bus. It is used in a wide variety of computer systems sold today ranging from laptops to large servers. Its bandwidth and efficient support for multiple masters has allowed it to sustain high performance applications while at the same time, its low pin count and high integration factor has enabled very low cost solutions.

Power Management in the current PC platform is performed by a combination of BIOS and System Management Mode (SMM) code utilizing hardware unique to each platform. While this strategy has successfully brought the PC platform into the mobile environment it is beset with problems because of the fact that there is no standard way to truly determine when the system is busy and when it is actually idle. The operating system does have this information so it makes sense to give it the responsibility for power management. The reason that this has not happened up to now is a lack of standards to provide the operating system with the required information that would allow it to control the hardware in a platform independent way. This specification addresses this need.

While the ***PCI Local Bus Specification*** is quite complete with a solid definition of protocols, electrical characteristics and mechanical form factors, no provision was made in the original specification for supporting power management functionality. This specification addresses this requirement by defining four distinct power states for the PCI bus and four distinct power states for PCI functions as well as an interface for controlling these power states.

3.1.1 Goals of this Specification

The goal of this specification is to establish a standard set of PCI peripheral power management hardware interfaces and behavioral policies. Once established this infrastructure enables an operating system to intelligently manage the power of PCI functions, and buses.

Detailed Goals for PCI Power Management Interface:

- Enable multiple PCI function power levels
- Establish a standard for PCI function wakeup events
- Establish a standard for reporting power management capabilities
- Establish a standard mechanism for controlling a PCI function's power state
- Establish a standard mechanism for controlling a PCI bus's power state
- Minimal impact to the ***PCI Local Bus Specification***
- Backwards compatible with ***PCI Local Bus Specification,*** Revision 2.1, and 2.0 compliant designs
- Preserve the designer’s ability to deliver differentiated products
- Provide a single architecture for all markets from mobile through server

Key Attributes of this Specification:

- Enhances the PCI Bus's Plug and Play capabilities by comprehending power management
- Standardized power state definitions
- Standardized register interface in PCI configuration space
- Standardized wake events

3.1.2 Target Audience

This document is intended to address the needs of developers of PCI-to-CardBus bridges. This document describes the hardware requirements of such devices to allow the power management of those devices in an operating system directed power management environment.

Software developers are also a targeted audience for this specification. Specifically, developers of operating systems and device drivers need to understand the power management interfaces presented by compliant devices to be able to manage them.

3.1.3 Overview/Scope

In order to implement a power managed system under the direction of the operating system, a large array of tightly coupled hardware, and software ingredients needs to be defined and integrated. The following diagram outlines, at a high level, this set of required architectural building blocks.

©2001 PCMCIA/JEITA

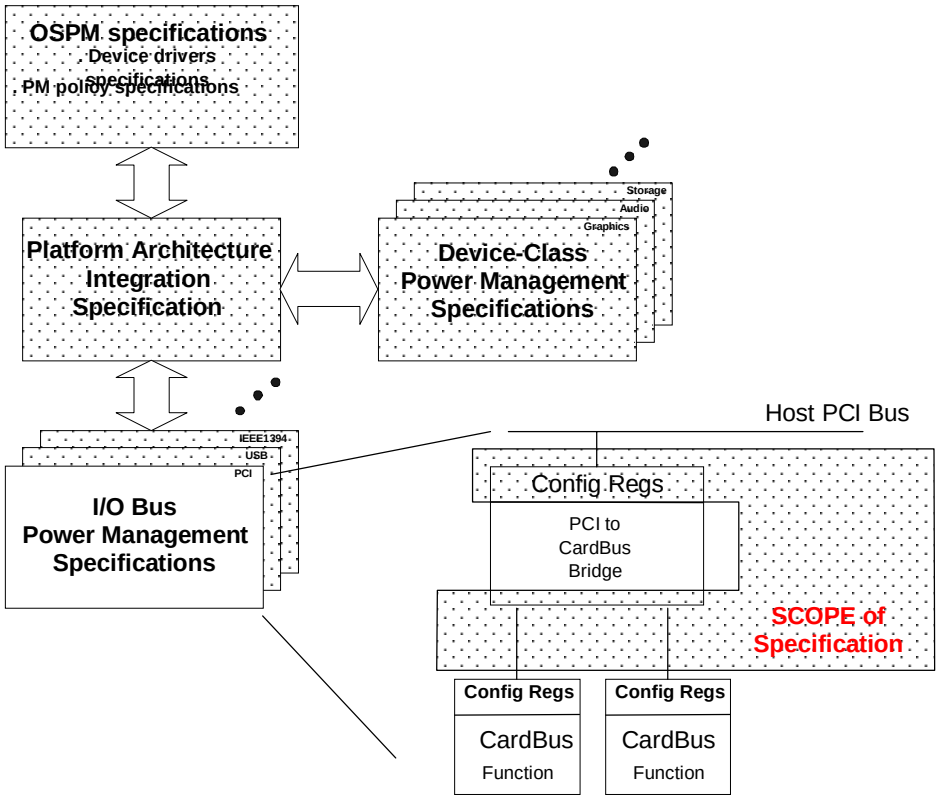


Figure 3-1: Operating System Directed Power Management System Architecture

The scope of this specification is sharply focused on establishing a standard set of interfaces that are required to power manage PCI-to-CardBus Bridges, buses, devices, CardBus cards and functions.

Devices which can only be implemented on the motherboard are power managed in motherboard specific ways (such as ACPI), and as such, may fall outside the scope of this specification. Docking bridges, for example fall into the motherboard devices category because the physical docking bridge component always resides logically on the motherboard, and is never deployed as an add-in card. As such Docking bridges are not covered by this specification.

The PCI Bus Power Management Interface Specification for CardBus describes requirements for implementing power management for PCI-to-CardBus bridge and CardBus card functions which are capable of being used on an add-in card.

3.1.4 Conventions Used in This Document

Several conventions are used in this document to help make it more readable. These are listed below.

- Power states are shown in bold italic text such as ***D0***.
- Register names are shown in bold text as in **PMCSR**.
- Names of bits or fields within registers are in italic text such as *PowerState.*
- Signal names are all capitalized and bold like **PME#**.

- All numbers are represented in decimal unless followed by a small letter.
- Hexadecimal numbers are represented with a following "H" (e.g. DCH).
- Binary numbers are represented with a following "B" (e.g. 10B).

3.2 PCI and CardBus Power Management Overview

3.2.1 PCI and CardBus Power Management States

Power management states are defined as varying, distinct levels of power savings. Power Management states are denoted by a state number. Throughout this document power management states for both PCI and CardBus buses and PCI and CardBus functions will be defined, specified and discussed. Power management states for PCI buses, by convention, are prefixed with a "***B***", and end in the power management state number (***0-3***). The higher the number the more aggressive the intended power savings. Similarly for PCI and CardBus functions the power management state is prefixed with a "***D***", and ends with the power management state number (***0-3***). Intended power savings increase with the power management state number.

3.2.1.1 PCI and CardBus Function Power States

Up to four power states are defined for each PCI and CardBus function in the system. These are ***D0-D3*** with ***D0*** being the maximum powered state, and ***D3*** being the minimum powered state. ***D1*** and ***D2*** power management states enable intermediate power savings states between the ***D0*** (on) and ***D3*** (off) power management states.

While the concept of these power states is universal for all functions in the system, the meaning, or intended functional behavior when transitioned to a given power management state is dependent upon the type (or class) of the function.

3.2.1.2 PCI and CardBus Bus Power States

The power management state of a bus can be characterized by certain attributes of the bus at a given time such as whether or not power is supplied, the speed of the clock, and what types of bus activities are allowed. These states are referred to as ***B0***, ***B1***, ***B2*** and ***B3***.

This specification defines a mechanism that enables explicit control of a PCI and CardBus bus's power and PCI and CardBus clock (**CCLK**) as a function the power management state of its originating device. The mechanism can be disabled (see ***3.3.2.5 PMCSR PCI-to-CardBus Bridge Support Extensions - PMCSR_BSE (Offset=6)***, and ***3.4.7.1 Control of Secondary Bus Power Source and Clock***).

3.2.1.3 Device-Class Specifications

The PCI and CardBus Bus Power Management Interface Specification standardizes the power management hardware interface for the PCI and CardBus Bus, and PCI and CardBus components.

However PCI and CardBus functions belonging to different device classes may behave differently when operating in the same power management state. This is the notion of Device-Class specific power management policy. For example, the list of capabilities that an audio subsystem would support in a given power management state would most likely be different than the list of capabilities supported by a graphics controller operating in the same power management state.

 ©2001 PCMCIA/JEITA

While Device-Class Power Management Specifications fall outside the scope of this specification, they are mentioned here to inform the reader of their important relationship to the interfaces defined in this document.

Each class of device must have its own class specific set of power management policies to define their intended behavior while in each of the power management states.

For a fully integrated power management system, these class-specific power management policies must also be standardized. Each major device type must have a "Device-Class Power Management Specification" that all manufacturers can use as a guide to facilitate the seamless integration of their power managed products into a system.

Device-Class Specifications will generally cover the following areas:

Device-class power characteristics	Each class of PCI and CardBus function should have a standard definition for each function power management state. This should include target power consumption levels, command response latencies, and state-change latencies. Implementation details for achieving these levels (such as whether an entire functional block is powered-off or the clock is stopped) might be important to a particular device class and, if so, should be specified. If any of these characteristics are in conflict with the requirements of the bus specifications, the function may not be able to implement that state on that particular bus.
Minimum Device-Class power capabilities	Each class of PCI and CardBus function should have a standard set of power capabilities appropriate to the class. An example might be to require support either for all four power states or for some lesser number. Requirements might also be specified for accuracy and frequency of power status updates. Finally, there might be class-specific requirements for Wakeup capabilities. For example, all modems should be able to wake the PC from ***D1***, and so on.
Device-class functional characteristics	Each class of PCI and CardBus function should have a standard definition of the available subset of functioning capabilities and features in each power state. For example, the network adapter can receive, but cannot transmit; the sound card is fully functional except that the power amps are off; and so on.
Device-class Wakeup characteristics	Each class of PCI and CardBus function should have a standard definition of its Wakeup policy, including a recommended resume latency specification. This includes specifying the various class-specific events that can wake up the system, and the power states from which the Wakeup can be signaled, with implementation details and examples where appropriate.

3.2.1.4 Bus Support for PCI and CardBus Function Power Management

Four base capabilities enable a robust power management solution. The capabilities are defined as:

Get Capabilities operation	This operation informs the operating system of the power management capabilities and features of a given function. It is performed as a part of the Operating System's device enumeration[1] and uses the information it receives to help determine the power management policies to implement in the system. Information required from the function include which power states are implemented, and the function's Wakeup capabilities.
Set Power State operation	This operation puts the function into a specific power management state and enables power management features based on the global system power management policy and the function's specific capabilities. This includes setting the function to wake the system from a sleeping state if certain events occur.
Get Power Status operation	This operation returns information about the current power state of the PCI function.
Wakeup operation	This is a mechanism for having a PCI or CardBus function wake the system from a sleeping state on specified events.

[1] The PCI function provides power management capabilities reporting through a standard register definition as specified in this document.

While individual PCI and CardBus functions must support only the first three capabilities with wakeup being optional, all basic power management operations must be supported by the bus architecture to ensure that PCI and CardBus functions on the bus can be power managed. The fourth operation, Wakeup, is not optional for a PCI-to-CardBus bridge device: these devices must support wakeup.

For multi-function PCI and CardBus components there is a common portion of bus interface logic that physically binds each of the supported functions to the PCI or CardBus bus. This common logic's power consumption is explicitly reported if supported, using the **Data** register of Function 0. For further detail on the optional reporting of PCI function power consumption see ***3.3.2.6 Data (Offset = 7)***.

Control of the common logic is handled by the multi-function device in a software transparent fashion. For further power savings in a runtime environment, the enabling and disabling of some portion of this common PCI or CardBus bus interface logic is the responsibility of the multi-function component hardware. This implicit power control, if implemented, may be based upon the power states of the functions behind it. As an example one might consider a hardware administered logic control policy where the common logic can not be internally powered down unless all of the functions hosted by the device have been placed in the $D3_{hot}$ state first.

3.3 PCI-to-CardBus Bridge Power Management Interface

The four basic power management operations that have been defined are: Capabilities Reporting, Power Status Reporting, Setting Power State and System Wakeup. Of these four capabilities all are required of each function with the exception of wakeup event generation. This section describes the format of the registers in PCI-to-CardBus bridge configuration space which are used by these operations.

The *Status* and Capabilities Pointer (*Cap_ptr*) fields have been highlighted to indicate where the PCI Power Management features appear in the standard Configuration Space Header.

©2001 PCMCIA/JEITA

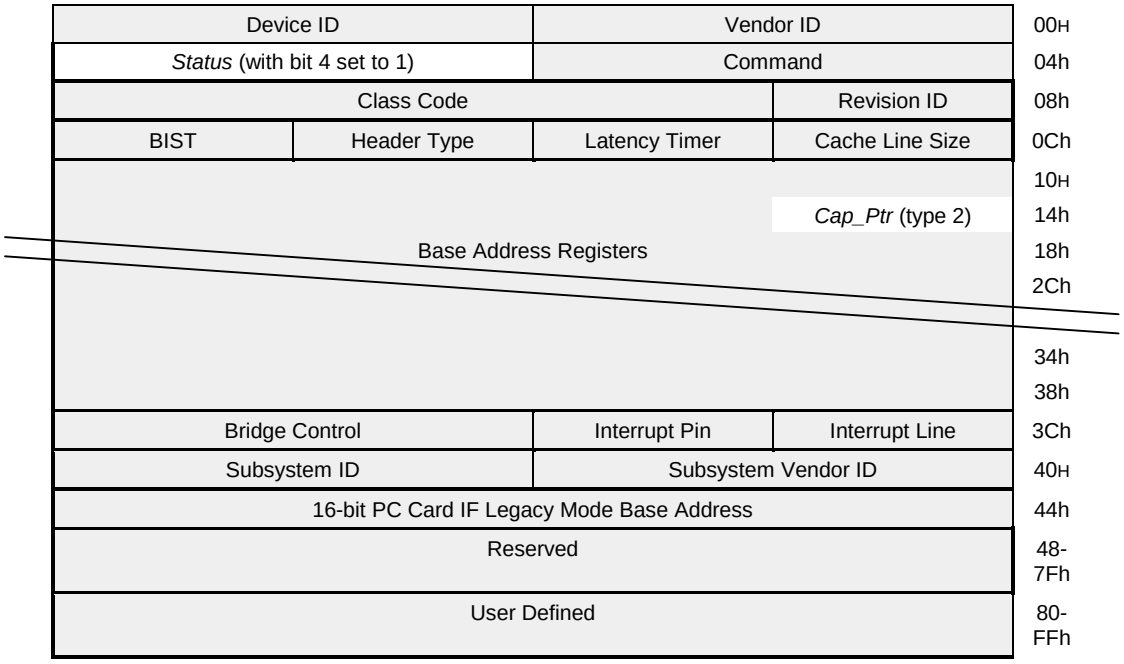


Figure 3-2: . Standard PCI Configuration Space Header Type 2

Software must have a standard method of determining if a specific function is designed in accordance with this specification. This is accomplished by using a bit in the PCI **Status** register to indicate the presence of the Capabilities List and a single byte in the standard PCI Configuration Space Header which acts as a pointer to a linked list of additional capabilities. Software can then traverse the list looking for the proper ID for PCI-to-CardBus Bridge Power Management (Cap_ID=01H). If there is no Capabilities List (*New Capabilities* bit in the **Status** register = 0) or if the list does not contain an item with the proper ID, the function does not support the PCI-to-CardBus Bridge Power Management interface described in this document and the Operating System will assume that the function only supports the ***D0*** and $\boldsymbol{D3_{cold}}$ power management states. These Legacy PCI-to-CardBus bridge functions may also require a device specific initialization sequence after any transition from $\boldsymbol{D3_{cold}}$ to ***D0*** (***see 3.8.3 Restoring PCI Functions From a Low Power State***).

3.3.1 Capabilities List Data Structure

The New Capabilities bit in the PCI **Status** Register (offset=06h) indicates whether or not the subject function implements a linked list of extended capabilities. Specifically, if bit 4 is set, the **Cap_Ptr** register is implemented to give the offset in configuration space for the first item in the list.

Table 3-1: PCI Status **Register**

Bits	Default Value	Read/Write	Description
15:05	--	--	Definition given in ***PCI Local Bus Specification Revision 2.1***
04	1B	Read Only	*New Capabilities* - This bit indicates whether this function implements a list of extended capabilities such as PCI Power Management. When set this bit indicates the presence of New Capabilities. A value of 0 means that this function does not implement New Capabilities.
03:00	0H	Read Only	Reserved

The location of the Capabilities Pointer (**Cap_Ptr**) depends on the PCI-to-CardBus bridge header type. See ***3.3.1.1 Capabilities List Cap_Ptr Location*** for Header Type specific Cap_Ptr offsets.

Table 3-2: Capabilities Pointer - Cap_Ptr

Bits	Default Value	Read/Write	Description
07:00	XXH	Read Only	The **Cap_Ptr** provides an offset into the function's PCI Configuration Space for the location of the first item in the Capabilities Linked List. The **Cap_Ptr** offset is DWORD aligned so the two least significant bits are always "0"s.

If a function does not implement any capabilities with IDs defined by the PCI SIG, the *New Capabilities* bit in the PCI **Status** register (bit 4) should read as "0" and the **Cap_Ptr** register should be ignored. Values of 00H-7FH are not valid values for the **Cap_Ptr** because they point into the standard PCI-to-CardBus bridge header. A PCI-to-CardBus bridge function may choose any DWORD aligned offset as indicated in ***Table 3-3: PCI Configuration Space Header Type / Cap_Ptr Mappings***.

©2001 PCMCIA/JEITA

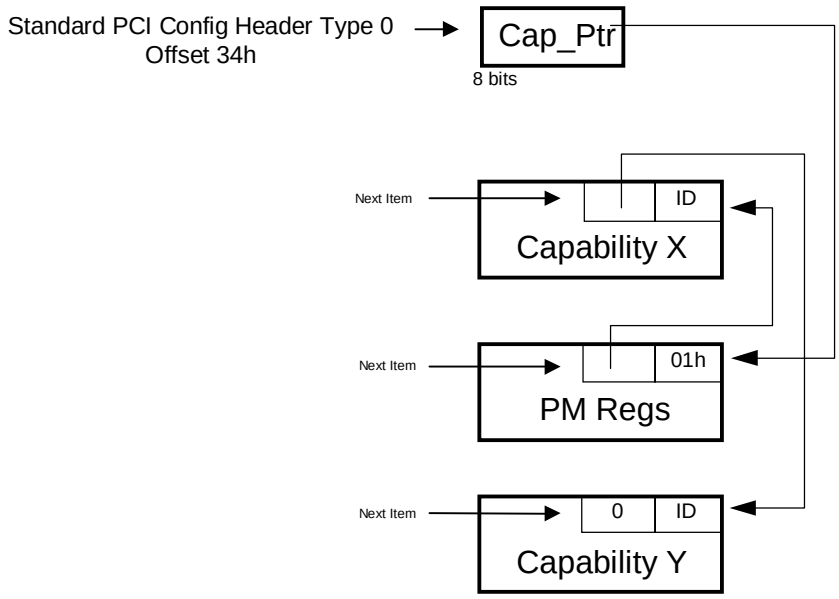


Figure 3-3: Capabilities Linked List

The figure above shows how the capabilities list is implemented. The **Cap_Ptr** gives the location of the first item in the list which is the PCI Power Management Registers in this example (although the capabilities can be in any order). The first byte of each entry is required to be the ID of that capability. The PCI Power Management capability has an ID of 01H. The next byte is a pointer giving an absolute offset in the functions PCI Configuration space to the next item in the list and must be DWORD aligned. If there are no more entries in the list, the **Next_Item_Ptr** must be set to 0 to indicate that the end of the linked list has been reached. Each capability can then have registers following the **Next_Item_Ptr**. The definition of these registers (including layout, size and bit definitions) is specific to each capability. The PCI Power Management Register Block is defined in this specification.

3.3.1.1 Capabilities List Cap_Ptr Location

There are currently three defined PCI Configuration Space Header types. The following table shows where to find the **Cap_Ptr** register for each of these Header Types.

Table 3-3: PCI Configuration Space Header Type / Cap_Ptr **Mappings**

Header Type	**Associated PCI Function Type**	Cap_Ptr **(PCI Config Space Offset)**	**Minimum Value**	**Maximum Value**
0	All other	34H	040H	0F8H
1	PCI to PCI Bridge	34H	040H	0F8H
2	PCI-to-CardBus Bridge	14H	080H	0F8H

Regardless of the implemented Header Type the definition of the **Cap_Ptr** register and *the New Capabilities* bit in the PCI **Status** register is common.

3.3.2 Power Management Register Block Definition

This section describes the PCI Power Management Interface registers.

The figure below illustrates the organization of the PCI Power Management Register Block. The first 16 bits (Capabilities ID [offset = 0] and Next Item Pointer [offset = 1]) are used for the linked list infrastructure.

The next 32 bits (**PMC** [offset = 2] and **PMCSR** registers [offset = 4]) are required for compliance with this specification. The next 8 bit register (Bridge support **PMCSR** extensions [offset = 6]) is required only for bridge functions, and the remaining 8 bit **Data** register [offset = 8] is optional for any class of function. As with all PCI configuration registers, these registers may be accessed as bytes, 16 bit words or 32 bit DWORDs.

Unless otherwise specified, all write operations to reserved registers must be treated as no-ops; that is, the access must be completed normally on the bus and the data is discarded. Read accesses to reserved or unimplemented registers must be completed normally and a data value of 0 returned.

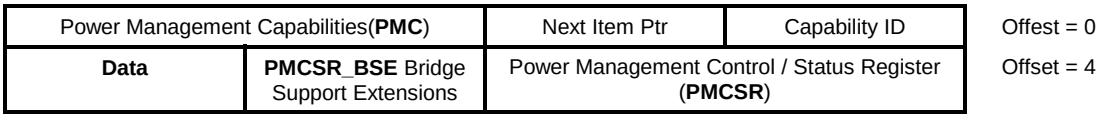


Figure 3-4: Power Management Register Block

The offset for each register is listed as an offset from the beginning of the linked list item which is determined either from the **Cap_Ptr** (If Power Management is the first item in the list) or the **Next_Item_Ptr** of the previous item in the list.

3.3.2.1 Capability Identifier - Cap_ID (Offset = 0)

The Capability Identifier, when read by system software as 01H indicates that the data structure currently being pointed to is the PCI Power Management data structure. Each function of a PCI device may have only one item in its capability list with **Cap_ID** set to 01H.

Table 3-4: Capability Identifier - Cap_ID

Bits	Default Value	Read/Write	Description
07:00	01H	Read Only	**ID** - This field, when "01H" identifies the linked list item as being the PCI Power Management Registers.

©2001 PCMCIA/JEITA

3.3.2.2 Next Item Pointer - Next_Item_Ptr (Offset = 1)

The Next Item Pointer Register describes the location of the next item in the function's capability list. The value given is an offset into the function's PCI Configuration space. If the function does not implement any other capabilities defined by the PCI SIG for inclusion in the capabilities list, or if power management is the last item in the list, then this register must be set to 00H.

Table 3-5: Next Item Pointer - Next_Item_Ptr

Bits	Default Value	Read/Write	Description
07:00	00H	Read Only	**Next Item Pointer** - This field provides an offset into the function's PCI Configuration Space pointing to the location of next item in the function's capability list. If there are no additional items in the Capabilities List, this register is set to 00H.

3.3.2.3 Power Management Capabilities - PMC (Offset = 2)

The Power Management Capabilities register is a 16 bit read-only register which provides information on the capabilities of the function related to power management. The information in this register is generally static and known at design time.

Table 3-6: Power Management Capabilities - PMC

Bits	Default Value	Read/Write	Description
15	Implementation Specific	Read/Write	*PME_Support* - This bit field indicates the power states in which the function may assert **PME#**. A value of 0B for this bit indicates that the function is not capable of asserting the **PME#** signal while in that power state and that V_{AUX} is not supported. This bit is also part of the PCI-to-CardBus bridge device's PME context. PCI-to-CardBus bridge devices are required to support ***D3*$_{cold}$**. bit(15) 1XXXXB - **PME#** can be asserted from ***D3*$_{cold}$** (See note following)
14:11	Device Specific	Read Only	*PME_Support* - This four bit field indicates the power states in which the function may assert **PME#**. A value of 0B for any bit indicates that the function is not capable of asserting the **PME#** signal while in that power state. bit(11) XXXX1B - **PME#** can be asserted from ***D0*** bit(12) XXX1XB - **PME#** can be asserted from ***D1*** bit(13) XX1XXB - **PME#** can be asserted from ***D2*** bit(14) X1XXXB - **PME#** can be asserted from ***D3*$_{hot}$**
10	1	Read Only	*D2_Support* - If this bit is a "1", this function supports the ***D2*** Power Management State. PCI-to-CardBus bridge devices are required to support device state ***D2***.
09	1	Read Only	*D1_Support* - If this bit is a "1", this function supports the ***D1*** Power Management State. PCI-to-CardBus bridge devices are required to support device state ***D1***.
08:06	000B	Read Only	Reserved
05	0	Read Only	*DSI* - The Device Specific Initialization bit has no meaning for a PCI-to-CardBus bridge device and reads "0".
04	Device Specific	Read Only	*Auxiliary Power Source* (V_{AUX}) - When this bit is a "1" it indicates that support for **PME#** in ***D3*$_{cold}$** requires auxiliary power supplied by the system by way of a proprietary delivery vehicle. A "0" in this bit indicates that the function supplies its own auxiliary power source.
03	Device Specific	Read Only	*PME Clock* - When this bit is a "1" it indicates that the function relies on the presence of the PCI clock for **PME#** operation. When this bit is a "0" it indicates that no PCI clock is required for the function to generate **PME#**. Functions that do not support **PME#** generation in any state must return "0" for this field.
02:00	001B	Read Only	*Version* - A value of 001B indicates that this function complies with the Revision 1.0 of the ***PCI Power Management Interface Specification.***

Note: For a PCI-to-CardBus Bridge device, the setting of bit 15, **PME#** can be asserted from ***D3*$_{cold}$**, indicates that this system implementation has incorporated V_{AUX} to support ***D3*$_{cold}$** wakeup events through the bridge. This register bit should be programmed by the host platform BIOS at POST time to indicate the system implementation. Supporting V_{AUX} for the PCI-to-CardBus Bridge also indicates that the socket will supply up to 200 mA of Vcc current in the ***D3*$_{cold}$** state for CardBus card usage. The state of this bit is also a part of the PCI-to-CardBus bridge's PME context.

©2001 PCMCIA/JEITA

3.3.2.4 Power Management Control/Status - PMCSR (Offset = 4)

This 16 bit register is used to manage the PCI-to-CardBus bridge function's power management state as well as to enable / monitor power management events.

The Power Management Event support bits, *PME_Status*, and *PME_En*, are defined to be "sticky" bits for functions that can generate power management events from ***D3*$_{cold}$**, in that their states are not affected by power on reset or transitions from ***D3*$_{cold}$** to the ***D0*** Uninitialized state. Preservation of these bits is typically achieved by either powering them with an auxiliary power source, or by using non-volatile storage cells for them. The only way to clear out these bits is to have system software write to them with the appropriate values.

As mentioned previously, the PME Function Context is defined as the logic responsible for identifying power management events, the logic responsible for generating the **PME#** signal and the bits within this register that provide the standard system interface for this functionality. PME Function Context also contains any device class specific status that must survive the transition to the ***D0*** Uninitialized state as well.

If a function supports **PME#** generation from ***D3*$_{cold}$**, its PME Function Context is not affected by either a PCI Bus Segment Reset (hardware component reset), or the internal "soft" re-initialization that occurs when restoring a device from ***D3*$_{hot}$**. This is because the function's Power Management Event functionality itself may have been responsible for the wake event which caused the transition back to ***D0***. Therefore, the PME Function Context must be preserved for the system software to process.

If **PME#** generation is not supported from ***D3*$_{cold}$** then all PME Function Context is initialized with the assertion of a bus segment reset.

Because a PCI Bus **RST#** assertion does not necessarily clear all functions' PME Function Context, (functions that support **PME#** from ***D3*$_{cold}$**), the system software is required to explicitly initialize all PME Function Context, including the Power Management Event support bits, for all functions during initial operating system load. In terms of the **PMCSR** this means that during the initial operating system load each function's *PME_En* bit must be written with a "0", and each function's *PME_Status* bit must be written with a "1" by system software as part of the process of initializing the system.

Table 3-7: Power Management Control/Status - PMCSR

Bits	Value at Reset	Read/ Write	Description
15	Sticky Bit, indeterminate at time of initial OS boot if function supports **PME#** from ***D3*$_{cold}$**. 0B, if the function does not support **PME#** from ***D3*$_{cold}$**.	Read/ Wr-Clear	*PME_Status* - This bit is set when the PCI-to-CardBus bridge would normally assert the **PME#** signal independent of the state of the *PME_En* bit. Writing a "1" to this bit will clear it and cause the PCI-to-CardBus bridge to stop asserting a **PME#** (if enabled). Writing a "0" has no effect. This bit defaults to "0" if the function does not support **PME#** generation from ***D3*$_{cold}$**. Since the PCI-to-CardBus bridge function supports **PME#** from ***D3*$_{cold}$** this bit is sticky and must be explicitly cleared by the operating system each time the operating system is initially loaded unless a PME is outstanding.
14:13	Device Specific	Read Only	*Data_Scale* – This two bit read-only field indicates the scaling factor to be used when interpreting the value of the **Data** register. The value and meaning of this field will vary depending on which data value has been selected by the *Data_Select* field. This field is required for any function that implements the **Data** register. It is otherwise optional. See ***3.3.2.6 Data (Offset = 7)*** for more details.
12:09	0000B	Read/ Write	*Data_Select* – This four bit field is used to select which data is to be reported through the **Data** register and *Data_Scale* field. This field is required for any function that implements the **Data** register. See ***3.3.2.6 Data (Offset = 7)*** for more details.
08	Sticky Bit, indeterminate at time of initial OS boot if function supports **PME#** from ***D3*$_{cold}$**. 0B, if the function does not support **PME#** from ***D3*$_{cold}$**.	Read/ Write	*PME_En* - "1" enables the function to assert **PME#**. When "0" **PME#** assertion is disabled. This bit defaults to "0" if the function does not support **PME#** generation from ***D3*$_{cold}$**. If the function supports **PME#** from ***D3*$_{cold}$** then this bit is sticky and must be explicitly cleared by the operating system each time the operating system is initially loaded. In the event Vcc and V_{AUX} are removed, upon restoration of Vcc or V_{AUX}, *PME_En* and *PME_Status* are cleared.
07:02	000000B	Read Only	Reserved
01:00	00B	Read/ Write	*PowerState* - This two bit field is used both to determine the current power state of a function and to set the function into a new power state. The definition of the field values is given below. 00B - ***D0*** 01B - ***D1*** 10B - ***D2*** 11B - ***D3*$_{hot}$** If software attempts to write an unsupported, optional state to this field, the write operation must complete normally on the bus, however the data is discarded and no state change occurs.

3.3.2.5 PMCSR PCI-to-CardBus Bridge Support Extensions - PMCSR_BSE (Offset=6)

PMCSR_BSE supports PCI-to-CardBus bridge specific functionality and is required for all PCI-to-PCI and PCI-to-CardBus bridges.

 ©2001 PCMCIA/JEITA

Table 3-8: PMCSR Bridge Support Extensions - PMCSR_BSE

Bits	Value at Reset	Read/Write	Description
07	External strap or internally hardwired	Read Only	*BPCC_En* (Bus Power/Clock Control Enable) - "1" indicates that the bus power/clock control mechanism as defined in ***3.4.7.1 Control of Secondary Bus Power Source and Clock*** is enabled. A "0" indicates that the bus power/clock control policies defined in ***3.4.7.1*** have been disabled. When the Bus Power/Clock Control mechanism is disabled the bridge's **PMCSR** *PowerState* field cannot be used by the system software to control the power or clock of the bridge's secondary bus.
06	External strap or internally hardwired	Read Only	*B2_B3#* (***B2***/***B3*** support for $D3_{hot}$) - The state of this bit determines the action that is to occur as a direct result of programming the function to $D3_{hot}$. A "1" indicates that when the bridge function is programmed to $D3_{hot}$, its secondary bus's PCI clock will be stopped (***B2***). This bit is only meaningful if bit 7 (*BPCC_En*) is a "1". See ***3.4.7.1 Control of Secondary Bus Power Source and Clock*** for details.
05:00	000000B	Read Only	Reserved

3.3.2.6 Data **(Offset = 7)**

The **Data** Register is an optional 8 bit read-only register that provides a mechanism for the function to report state dependent operating data such as power consumed or heat dissipation. Typically, the data returned through the **Data** register is a static copy (look up table, for example) of the function's typical worst case "DC characteristics" data sheet.

Any type of data could be reported through this register, but only power usage is defined by this version of the specification. If the **Data** register is implemented then the *Data_Select* and *Data_Scale* fields must also be implemented. If this register is not implemented, a value of 0 should always be returned by this register as well as for the *Data_Select* and *Data_Scale* fields.

Table 3-9: Data **Register**

Bits	Default Value	Read/Write	Description
07:00	00H	Read Only	**Data** - This register is used to report the state dependent data requested by the *Data_Select* field. The value of this register is scaled by the value reported by the *Data_Scale* field.

The **Data** register is used by writing the proper value to the *Data_Select* field in the **PMCSR** and then reading the *Data_Scale* field and the **Data** register. The binary value read from **Data** is then multiplied by the scaling factor indicated by *Data_Scale* to arrive at the value for the desired measurement. The table below shows which measurements are defined and how to interpret the values of each register.

Table 3-10: Power Consumption/Dissipation Reporting

Value in *Data_Select*	Data Reported	*Data_Scale* Interpretation	Units/Accuracy
0	***D0*** Power Consumed	0 = Unknown 1 = 0.1x 2 = 0.01x 3 = 0.001x	Watts
1	***D1*** Power Consumed		
2	***D2*** Power Consumed		
3	***D3*** Power Consumed		
4	***D0*** Power Dissipated		
5	***D1*** Power Dissipated		
6	***D2*** Power Dissipated		
7	***D3*** Power Dissipated		
8	Common logic power consumption (Multi-function PCI devices, Function 0 only)		
9-15	Reserved (function 0 of a multi-function device)	0 = Unknown 1-3 = TBD	TBD
8-15	Reserved (single function PCI devices, and other functions (greater than function 0) within a multi-function device)	0 = Unknown 1-3 = TBD	TBD

When using the **Data** register as a window into the data sheet for the PCI function, data returned must comply with measurements derived from the following test environment:

- Bus Frequency: 33MHz / 66MHz (use 66MHz characterization if the function is 66MHz capable for the worst case data)
- Vcc: 5.25 VDC or 3.3 VDC (if 5 VDC not supported).

The power measurements defined above have a dynamic range of 0 to 25.5 W with 0.1 W resolution, 0 to 2.55 W with 0.01 W resolution or 0 to 255 mW with 1 mW resolution. Power should be reported as accurately as possible. For example, the data returned for each state supported must indicate the maximum power used by the function when in that particular state. The "Power Consumed" values defined above must include all power consumed from the PCI power planes through the PCI connector pins. If the PCI card provides power to external devices that power must be included as well. It should not include any power derived from a battery or an external source. This information is useful for management of the power supply or battery.

The "Power Dissipated" values provide the amount of heat which will be released into the interior of the computer chassis. This excludes any power delivered to external devices but must include any power derived from a battery or external power source and dissipated inside the computer chassis. This information is useful for fine grained thermal management.

If a function allows a wide range of implementation options, the values reported through this register may need to be loadable through a serial EPROM or strapping option at reset much like the Subsystem Vendor ID and Subsystem ID registers

Multi-function devices implementing power reporting should report the power consumed by each function in each corresponding function's Configuration Space. In a multi-function device, the common logic power consumption is reported in function 0's Configuration Space through the **Data** register once the *Data_Select* field of the function 0's **PMCSR** has been programmed to "1000B". The sum of the values reported should then be accurate for the condition of all functions in the device being put in that state.

©2001 PCMCIA/JEITA

Multi-device cards implementing power reporting (i.e. multiple PCI devices behind a PCI bridge) should have the bridge report the power it uses by itself. Each function of each device on the card is responsible for reporting the power consumed by that function.

3.4 PCI/CardBus Bus Power States

This section describes the different power states of the PCI / CardBus bus itself. The defined bus states are common possibilities for both the bridge ordinate and subordinate buses – that is the PCI-to-CardBus bridge can expect to support the defined bus states as either a target or provider.

From a power management perspective, the PCI Bus can be characterized at any point in time by one of four power management states. ***B0*** corresponds to the bus being fully useable (full power, and clock frequency) and ***B3*** meaning that the power to the bus has been switched off. ***B1***, and ***B2*** represent intermediate power management states. The ***B1*** bus power management state is defined as a fully powered yet "enforced" idle[2] PCI bus with its clock free running. The ***B2*** state carries forward the characteristics of the ***B1*** state, but also has its clock stopped.

A PCI-to-CardBus bridge device must be capable of providing to the CardBus bus the defined bus states and be capable of supporting each of the defined bus states on its primary or ordinate bus. Bus state ***B2*** can be provided to the CardBus bus using Mobile PCI's clock run protocol.

The table below shows a mapping of the four defined power states to key characteristics of the PCI and CardBus bus.

Table 3-11. PCI/CardBus Bus Power Management States

PCI/CardBus Bus States	Vcc	Clock	Bus Activity
B0 (Fully On)	On	Free running, ***PCI Local Bus Specification***, Revision 2.1 and CardBus compliant	Any PCI or CardBus Transaction, Function Interrupt, or PME Event
B1	On	Free running, ***PCI Local Bus Specification***, Revision 2.1 and CardBus compliant	PME Event
B2	On	Off or clock run asserted	PME Event
B3	May be off	Off or clock run asserted.	PME Event

Each PCI or CardBus bus in a system has an originating device which can support one or more power states. In the case of CardBus, this will be a PCI-to-CardBus bridge.

3.4.1 PCI/CardBus *B0* State - Fully On

All buses support ***B0*** by default.

A bus in ***B0*** is capable of running any legal PCI or CardBus transaction.

Since ***B0*** is the only PCI / CardBus Bus power management state where data transactions can take place, system software must ensure that a PCI / CardBus bus is in ***B0*** before attempting to access any PCI / CardBus resources on that bus. If an access is attempted to a function residing downstream of a

[2] Enforced by the operating system which has previously programmed the functions residing on, and further downstream of, that particular bus segment to power management states that would preclude normal PCI transactions or functional interrupts from occurring.

bus that is not in ***B0***, the transaction must be treated as if a "Master Abort" had occurred and error reporting handled in the same manner.

It is the system software's responsibility to ensure that, prior to attempting to program the CardBus bus to a power management state other than ***B0,*** all functions residing on the CardBus card have previously been programmed to a state that would preclude any further bus activity initiated by them. For new CardBus functions compliant with the PCI/CardBus Bus Power Management Interface Specification this means that all CardBus card functions must have been previously programmed to a power management state that, for each of them, has precluded any bus activity on their parts. For legacy PCI/CardBus functions system software must rely on other means such as disabling the Bus Master Enable bit of the legacy function's PCI/CardBus Command register to ensure that the function does not attempt to initiate any bus transactions.

The bus's originating device must always exit from ***B0*** gracefully by first allowing the bus to settle into the idle state.

3.4.2 PCI/CardBus *B1* State

When a PCI or CardBus bus is in ***B1***, Vcc is still applied to all devices on the bus however no bus transactions are allowed to take place on the bus except type 0 configuration cycles.

When the PCI-to-CardBus bridge is put in state ***D1***, the subordinate CardBus bus will go to bus state ***B1*** and the CardBus bus will be idle with clock running unless clock run protocol is asserted.

The PCI-to-CardBus bridge controller will only respond to PCI type 0 configuration cycles intended for the PCI-to-CardBus bridge device. Attempts to access devices on the PCI-to-CardBus bridge subordinate buses will not be claimed by the PCI-to-CardBus bridge controller.

The ***B1*** state provides the bridge with information indicating that no functions residing on the bus will attempt to initiate any bus transactions or functional or Card Status Change interrupts, with the possible exception of a power management event which goes through the PCI-to-CardBus bridge. This information can be used to intelligently apply more aggressive power savings in the bridge design.

3.4.3 PCI/CardBus *B2* State

When a PCI/CardBus bus is in ***B2***, Vcc is still applied to all devices on the bus but the clock is stopped, and held in the low state. All PCI/CardBus Bus signals are required to be held at valid logic states at all times, consistent with the ***PCI Local Bus Specification Revision 2.1***.

The ***B2*** state provides the bridge function with information indicating that no functions residing on the bus will attempt to initiate any bus transactions, functional or Card Status Change interrupts, with the possible exception of a power management event. Nor do any of the PCI/CardBus functions require a PCI clock. This information could then be used to intelligently apply more aggressive power savings in the bridge design. There is a minimum time requirement of 50 ms which must be provided by system software between when the bus is switched from ***B2*** to ***B0*** and when a device on the bus is accessed to allow time for the clock to start up and the bus to settle.

CardBus bus clock stop can be accomplished using Mobile PCI's clock run protocol.

3.4.4 PCI/CardBus *B3* State - Off

In ***B3,*** Vcc may have been removed from all devices and the PCI-to-CardBus bridge may be operating on V_{AUX}. When Vcc is reapplied to the PCI/CardBus bus, **RST#** must be asserted for that bus

©2001 PCMCIA/JEITA

segment, and the bus brought to an active, idle state in accordance with ***the PCI Local Bus Specification, Revision 2.1*** and the ***Electrical Specification***.

In the case of a PCI-to-CardBus bridge in the $D3_{cold}$ state, after the bridge controller receives its bus segment reset (**RST#**), the bridge controller must assert **CRST#** / **RESET** for its subordinate bus segments for slots having Vcc applied. Note that the bridge controller may not cause slot Vcc settings to change when returning to ***D0*** from any power state except on a card removal event.

B3 is exhibited by all ***PCI Local Bus Specification, Revision 2.1*** and ***PC Card Standard*** compliant systems when their power is removed. A programmable interface for placing a bus in ***B3***, or restoring it from ***B3*** is optional.

3.4.5 PCI Bus Power State Transitions

The PCI Bus Power States can be changed as shown in the figure below.

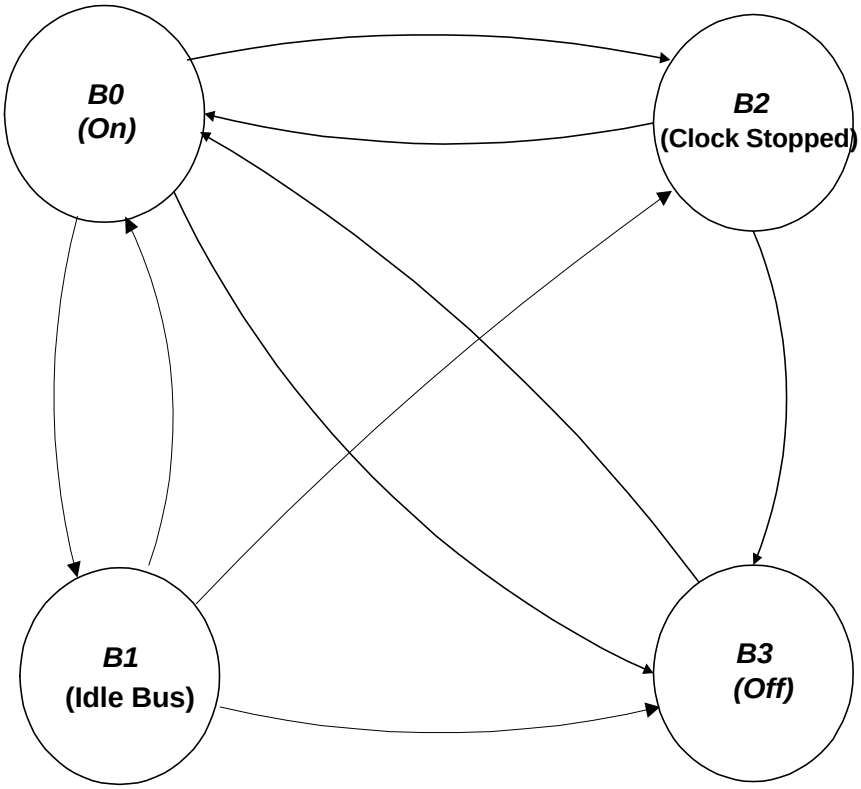


Figure 3-5: PCI Bus PM State Transitions

A system reset always returns the PCI bus to ***B0***.

Removing power always takes the bus to ***B3***. All other bus state changes are made by software, which writes to the appropriate location in the bus's originating device. These programmed function power state transitions implicitly have impact on the next power state for a particular bus. All buses support ***B3*** by default if power is removed from the system. A bridge may optionally support ***B3*** when its power state is programmed to $D3_{hot}$

3.4.6 PCI Clocking Considerations

PCI bridge functions, must either directly drive and control the clock to their secondary bus or provide sideband information to an external clock source for that bus segment. Mobile PCI Clock Run protocol also meets this requirement.

Unless otherwise specified, all PCI and CardBus bus clocking is required to be ***PCI Local Bus Specification, Revision 2.1*** and ***PC Card Standard Electrical Specification*** compliant.

3.4.7 Control/Status of PCI and CardBus Bus Power Management States

PCI bus power management states, as defined in this specification, adhere to the general policy that a bus's power state follows, or tracks, that of its originating device's power management state. So by writing to, or reading from, an originating bridge function's **PMCSR** *PowerState* field the operating system can explicitly set, or determine its PCI bus's power management state.

Behavioral policy for power managed PCI (the PCI-to-CardBus bridge) and CardBus functions on a given bus, as defined in ***3.5.6 PCI-to-CardBus Bridge Power Management Policies***, dictates that a PCI function must be at the same or greater power management state as that of the bus it physically resides on. For example, a PCI-to-CardBus bridge whose secondary bus is in ***B1*** could correctly assume that no CardBus functions on its secondary bus will attempt to initiate any bus traffic until the bridge's state is changed to ***D0*** which would result in the secondary bus's transition to ***B0***[3]**.** New PCI-to-CardBus bridge designs could take advantage of this information regarding its surroundings to potentially achieve further power savings in their designs.

3.4.7.1 Control of Secondary Bus Power Source and Clock

This section defines the standard mechanism that system software uses to control the clock, and power source of a PCI bus. This mechanism ties control of secondary bus power and clock to the originating device's power management state.

The following table defines the relationship between an originating PCI bridge function's power management state, and that of its secondary bus. The third column defines actions that must occur as a direct consequence of the originating device's *PowerState* field having been programmed to the current power management state.

[3] Bus activity wouldn't actually resume until the downstream functions were also programmed to states that permitted bus transactions to occur.

©2001 PCMCIA/JEITA

Table 3-12: PCI Bus Power and Clock Control

Originating Device's Bridge PM State	Secondary Bus PM States	Resultant Actions by Bridge (either direct or indirect)
D0	***B0***	none
D1	***B1***	none
D2	***B2***	Clock stopped on secondary bus: Mobile PCI clock run protocol may be used in stopping the PCI-to-CardBus bridge's CardBus bus clock
***D3*$_{hot}$**	***B3***	Clock stopped and Vcc may be removed from the slot. (See definition of *B2_B3#* in ***Table 3-8: PMCSR Bridge Support Extensions - PMCSR_BSE***.)
***D3*$_{cold}$**	***B3***	V_{AUX} support only **CCLK** stopped low **CRST#** low

Note that the power management state of a PCI or CardBus bus segment follows that of its originating bridge's power management state with one exception. The ***D3*$_{hot}$** state may cause its secondary bus's power management state to transition to either ***B2*** or ***B3***.

If a system designer choose not to implement a programmed ***B3*** bus power management state when the bridge function is transitioned to ***D3*$_{hot}$**, the ***B2*** option allows the system to keep power applied to the bus segment while still achieving significant power savings. (idle bus with clock stopped)

Two single bit read only fields defined in the function's **PMCSR_BSE** register support this functionality. The *B2_B3#* bit is used to determine the power management state of the secondary bus when its originating device's bridge function is transitioned to ***D3*$_{hot}$**.

The *BPCC_En* (Bus Power / Clock Control Enable) bit**,** serves as an enable / disable bit supported to allow other, possibly system specific (ACPI, Hot-Plug, etc.) bus power and clock control schemes to bear the responsibility for controlling power and clock for a given PCI bus.

Clock control for the PCI-to-CardBus bridge's CardBus bus can be accomplished using the Mobile PCI clock run protocol. When the PCI-to-CardBus bridge desires to stop the clock to the CardBus card, it requests permission from the card. If the card supports stopping the clock, permission is granted as applicable. When the CardBus card requires that the clock be started again, it uses the clock run protocol to request that the CardBus clock be started from the PCI-to-CardBus bridge. This describes the ***D0*** / ***B0*** scenario. In CardBus card device states ***D1*** and ***D2***, the CardBus card can still request that the clock be started only if *PME_EN* is true. In device state ***D2***, the CardBus card must be capable of the clock run request being denied because the PCI-to-CardBus bridge may not have a PCI clock to pass on to the CardBus bus. It is expected that the PCI-to-CardBus bridge use the clock run protocol to stop the clock in when the PCI-to-CardBus bridge is placed in device state ***D2*** / ***B2***. The clock run protocol cannot be exercised in device state ***D3***.

3.5 PCI-to-CardBus Bridge Power Management States

PCI and CardBus define a device as a physical load on the PCI or CardBus bus. Each PCI or CardBus device can host multiple functions, each with its own PCI configuration space. Since each PCI or CardBus function is an independent entity to the software, each function must implement its own power management interface. Each PCI and CardBus function can be in one of four power management states. All PCI and CardBus functions that adhere to this specification are required to support ***D0***, ***D3*$_{hot}$**, and ***D3*$_{cold}$**.

D1 and ***D2*** are optional power management states for CardBus cards, but are required states for PCI-to-CardBus bridge devices. These intermediate states are intended to afford the system designer more flexibility in balancing power savings, restore time, and low power feature availability tradeoffs for a given device class. As the only PCI bridge that has functionality and **PME#** support between the PCI/CardBus device or function, the PCI-to-CardBus bridge device must not be the limiting factor in the PME chain. The ***D1*** state could, for example, be supported as a slightly more power consuming state than ***D2***, however one that yields more available features and quicker restore time than could be realized from ***D2***.

The ***D3*** power management state constitutes a special category of power management state in that a CardBus card function could be transitioned into ***D3*** either by software, or by physically removing power using the PCI-to-CardBus bridge device. In that sense the two ***D3*** variants have been designated as $D3_{hot}$ and $D3_{cold}$ where the subscript refers to the presence or absence of Vcc respectively. Functions in $D3_{hot}$ can be transitioned to an uninitialized ***D0*** state via software by writing to the function's *PMCSR* register or by having it's Bus Segment Reset (PCI **RST#**, CardBus card **CRST#**) asserted. PCI-to-CardBus bridge devices in the $D3_{cold}$ state can only be transitioned to an uninitialized ***D0*** state by reapplying Vcc and asserting Bus Segment Reset (**RST#**)

PCI-to-CardBus bridge functions operating in either ***D0***, ***D1***, ***D2***, or $D3_{hot}$ are required to be compliant with the ***PCI Local Bus Specification, Revision 2.1*** and the ***PC Card Standard Electrical Specification***.

3.5.1 PCI-to-CardBus Bridge *D0* State

All PCI-to-CardBus bridge and CardBus card functions must support the ***D0*** state.

A PCI-to-CardBus bridge device must initially be put into ***D0*** before being used. Upon entering ***D0*** from power on reset, or transition from $D3_{hot}$, the device will be in an uninitialized state. Once initialized by the system software the function will be in the ***D0*** active state. All PCI-to-CardBus bridge devices must support ***D0*** and a reset will force all PCI-to-CardBus bridge functions to the uninitialized ***D0*** state unless *PME_En* is true in which case PME context is retained. Legacy PCI-to-CardBus bridge functions built prior to the PCI/CardBus Bus Power Management Interface specification are assumed to be in ***D0*** whenever power is applied to them.

All PCI Configuration Space is fully functional.

3.5.2 PCI-to-CardBus Bridge *D1* State

Implementation of the ***D1*** power management state is required.

D1 is used as a light sleep state. Some functions may be processing background tasks such as monitoring the network which actually requires most of the function to be active. Allowable behavior for a given function in ***D1*** is dictated by the Device-Class-Power Management Specifications for that class of function.

All PCI Configuration Space is functional.

3.5.3 PCI-to-CardBus Bridge *D2* State

Implementation of the ***D2*** power management state is required.

When a PCI-to-CardBus bridge and it associated cards is not currently being used and probably will not be used for some time, it may be put into ***D2***. This state requires the function to provide significant power savings while still retaining the ability to fully recover to its previous condition. In this state the only PCI bus operation the PCI-to-CardBus bridge is allowed to initiate is a power

 ©2001 PCMCIA/JEITA

management event (PME). The function is only required to respond to PCI configuration accesses (i.e. memory and I/O spaces are disabled). Configuration Space must be accessible by system software while the function is in ***D2***.

System software must restore the function to ***D0*** active before memory or I/O space can be accessed. Initiated actions such as bus mastering and functional interrupt request generation can only commence after the function has been restored to an active state.

There is a minimum recovery time requirement of 200 µs between when a function is programmed from ***D2*** to ***D0*** and when the function can be next accessed as a target (including PCI Configuration Accesses). If an access is attempted in violation of the specified minimum recovery time, undefined system behavior may result.

All PCI Configuration Space is functional.

3.5.4 PCI-to-CardBus Bridge *D3* State

All PCI-to-CardBus bridges must support ***D3***.

In this state function context need not be maintained. However if power management events (PME) are supported from ***D3*** then PME context must be retained at a minimum if *PME_En* is true. When the function is brought back to ***D0*** (the only legal state transition from ***D3***) software will need to perform a full reinitialization of the function including its PCI configuration space.

There is a minimum recovery time requirement of 10 ms (enforced by system software) between when a function is programmed from ***D3*** to ***D0*** and when the function is accessed (including PCI Configuration Accesses). This allows time for the function to reset itself and bring itself to a power-on condition. It is important to note that regardless of whether the function is transitioned to ***D0*** from $\boldsymbol{D3_{hot}}$ or $\boldsymbol{D3_{cold}}$, the end result from a software perspective is that the function will be in the ***D0*** Uninitialized state.

All PCI Configuration Space can be read. The *PowerState*, *PME_En* and *PME_Status* bits must be functional.

3.5.4.1 Software Accessible *D3* (*D3*$_{hot}$)

PCI-to-CardBus bridges in $\boldsymbol{D3_{hot}}$ must respond to configuration space accesses as long as power and clock are supplied so that they can be returned to ***D0*** by software.

When programmed to ***D0*** the function performs the equivalent of a warm (soft) reset internally, and returns to the ***D0*** Uninitialized state without PCI **RST#** being asserted and without asserting **CRST#**/**RESET**. Other bus activity may be taking place during this time on the same PCI bus segment so the device that has returned to ***D0*** Uninitialized state must ensure that all of its PCI and CardBus bus signal drivers remain disabled for the duration of the $\boldsymbol{D3_{hot}}$ to ***D0*** Uninitialized state transition[4].

The only function context that must be retained in $\boldsymbol{D3_{hot}}$ and through the soft reset transition to the ***D0*** Uninitialized state is the PME context.

All PCI Configuration Space can be read and is valid. The *PowerState*, *PME_En* and *PME_Status* bits must be functional.

[4] PCI Bus signal drivers must behave the same as if the component had received a bus segment reset (**RST#**).

3.5.4.2 Power Off (*D3*$_{cold}$)

If Vcc is removed from a PCI-to-CardBus bridge device, all of its functions transition immediately to ***D3*$_{cold}$**. All devices support this state by default. When power is restored, PCI **RST#** must be asserted and functions will return to ***D0*** (***D0*** Uninitialized state) with a full ***PCI 2.1*** compliant power-on reset sequence. Whenever the transition from ***D3*** to ***D0*** is initiated through assertion of PCI **RST#**, the power-on defaults will be restored to the function by hardware just as at initial power up. The function must then be fully initialized and reconfigured by software after making the transition to the ***D0*** uninitialized state.

PCI-to-CardBus bridge devices that support power management events from ***D3*$_{cold}$** must preserve their PME context through the ***D3*$_{cold}$** to ***D0*** transition. The power required to do this must be provided by some auxiliary power source assuming that no power is made available to the PCI or CardBus device from the normal Vcc power plane. Note that for the PCI-to-CardBus bridge, PME context includes the mask / event registers typical of an Intel 82365 PC Card controller or the CardBus Socket Registers (dependent on what technology of card is installed) and bit 15, *PME_Support* for ***D3*$_{cold}$**. PCI-to-CardBus bridge controllers are required to support ***D3*$_{cold}$** with V_{AUX} support. I_{AUX} is limited to 10 µa. In the ***D3*$_{cold}$** state with V_{AUX} present, the PCI-to-CardBus Bridge device is required to turn off slot Vcc when a card is removed. When a card is removed, the I_{AUX} is allowed to peak to 5 ma for 15 ms in order to switch off Vcc to the appropriate socket. The card removal event may or may not also be a power management event. V_{AUX} support is further defined in ***3.7.2 Auxiliary Power for D3cold Power Management Events***.

3.5.5 PCI-to-CardBus Bridge Power State Transitions

All PCI-to-CardBus bridge and CardBus card function power management state changes are explicitly controlled by software except for hardware reset which brings all functions to the ***D0*** Uninitialized state. The figure and table below shows all supported state transitions. The unlabeled arcs represent a software initiated state transition (Set Power State Operation).

©2001 PCMCIA/JEITA

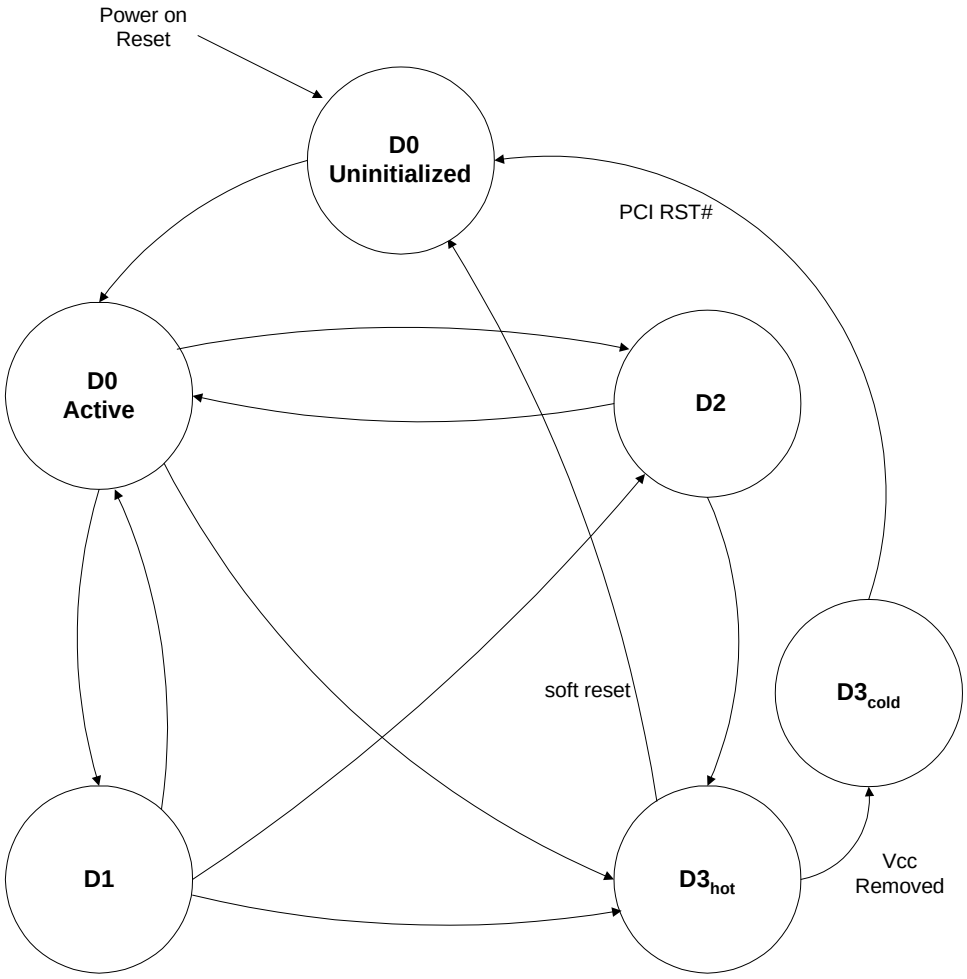


Figure 3-6: PCI Function Power Management State Transitions

Table 3-13: State Diagram Summary

Present state	*Valid next states*	PCI-to-CardBus Bridge Requirements in Present State
D0 (unitialized)	Software: • ***D0*** (active) • ***D3*** Hardware: • ***D0*** uninitialized via bus segment reset • Off	Subordinate bus: Vcc: Off; no change in applied Vcc if coming from $D3_{cold}$ under control of PCI-to-CardBus Bridge V_{AUX} and *PME_En* is true unless card is removed Bus: Per ***PC Card Standard*** for appropriate card technology Ordinate bus: ***B0*** PCI clock is running unless clock run is asserted. If clock run is not functional, clock is running. PCI bus is fully functional PCI (and ISA if implemented) interrupts are not functional PCI-to-CardBus Bridge responsibilities: Context (PCI bus in ***B0***) *PME_En* false Card detect pins valid, card type valid, slot Vcc off, context as programmed by BIOS or power on defaults *PME_En* true PME context; no change in Vcc to slot unless card is removed Bridge responds appropriately to configuration cycles for this PCI-to-CardBus Bridge device Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D0*** uninitialized state

Present state	Valid next states	PCI-to-CardBus Bridge Requirements in Present State
D0 (active)	Software: • ***D1*** • ***D2*** • ***D3*** Hardware: • ***D0*** uninitialized via bus segment reset • Off	Subordinate bus: ***B0*** CardBus clock is running unless Clock Run is asserted. If clock run is not functional in the PCI-to-CardBus bridge, clock is on CardBus bus is fully functional CardBus card functional and CardBusStatusChange interrupts are functional PCI-to-CardBus Bridge, CardBus and 16-bit PC Card **PME#** is available if *PME_En* is true CardBus card must be in ***D0***, ***D1***, ***D2*** or ***D3*** Slot Vcc is not changed unless card is removed Ordinate bus: ***B0*** PCI clock is running unless clock run is asserted. If clock run is not functional, clock is running. PCI bus is fully functional PCI (and ISA if implemented) interrupts are functional PCI-to-CardBus Bridge responsibilities: All context, PCI and functional, are available. Bridge responds appropriately to all PCI cycles for this PCI-to-CardBus Bridge device, its 16-bit PC Card and CardBus cards Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D0***/***B0*** state

©2001 PCMCIA/JEITA

Present state	Valid next states	PCI-to-CardBus Bridge Requirements in Present State
D1	*Software:* • ***D0*** • ***D2*** • ***D3*** *Hardware:* • ***D0*** uninitialized via bus segment reset • Off	Subordinate bus: ***B1*** CardBus clock is running unless Clock Run is asserted. If clock run is not functional in the PCI-to-CardBus bridge, clock is on CardBus bus is idle CardBus card functional and CardBusStatusChange interrupts are not available PCI-to-CardBus Bridge and CardBus **PME#** is available if *PME_En* is true CardBus card must be in ***D1***, ***D2*** or ***D3*** Card removal detection is functional Slot Vcc is not changed unless card is removed Ordinate bus: (***B0***) ***B1***: PCI clock is running unless clock run is asserted. If clock run is not functional, clock is running. PCI bus may be idle PCI (and ISA if implemented) interrupts are not available PCI-to-CardBus Bridge responsibilities: All context, PCI and functional, is preserved; PCI-to-CardBus Bridge PCI configuration space is functional; functional context is not available Bridge only responds to PCI type 0 cycles for this PCI-to-CardBus Bridge device Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D1***/***B1*** state but must be lower than the ***D0***/***B0*** state

Present state	*Valid next states*	PCI-to-CardBus Bridge Requirements in Present State
D2	Software: • ***D0*** • ***D3*** *Hardware:* • ***D0*** *uninitialized via bus segment reset* • *Off*	Subordinate bus: ***B2*** CardBus clock is not running unless Clock Run is required by CardBus card. If clock run is not functional in the PCI-to-CardBus bridge, clock is off CardBus bus is idle CardBus card functional and CardBusStatusChange interrupts are not available PCI-to-CardBus Bridge and CardBus **PME#** is available if *PME_En* is true CardBus card must be in ***D2*** or ***D3*** Card removal detection is functional Slot Vcc is not changed unless card is removed Ordinate bus: (***B0***, ***B1***) ***B2***: PCI clock is not running unless clock run is required. If clock run is not functional, clock is off. PCI bus is idle PCI (and ISA if implemented) interrupts are not available PCI-to-CardBus Bridge responsibilities: All context, PCI and functional, is preserved; PCI-to-CardBus Bridge PCI configuration space is functional; functional context is not available Bridge only responds to PCI type 0 cycles for this PCI-to-CardBus Bridge device Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D2*** / ***B2*** state but must be lower than the ***D1*** / ***B1*** state

Present state	Valid next states	PCI-to-CardBus Bridge Requirements in Present State
***D3*$_{hot}$**	Software: • ***D0*** uninitialized Hardware: • ***D0*** uninitialized via bus segment reset • ***D3*$_{cold}$** if V_{AUX} supplied • Off	Subordinate bus: ***B3*** CardBus clock is off CardBus bus is indeterminate except **CRST#** and **CCLK** must be low Interrupts are not available PCI-to-CardBus Bridge and CardBus **PME#** is available if *PME_En* is true CardBus card must be in ***D3*** Card removal detection is functional Slot Vcc is not changed unless card is removed Ordinate bus: (***B0***, ***B1***, ***B2***) ***B3***: PCI clock off PCI bus is off Interrupts are not available PCI-to-CardBus Bridge responsibilities: Only PME context is preserved if *PME_En* is true; PCI-to-CardBus Bridge PCI configuration space is readable; functional PME context is not available; only PCI-to-CardBus Bridge Power Management Control and Status Register *PowerState*, *PME_En* and *PME_Status* bits are required to be functional Bridge only responds to PCI type 0 cycles for this PCI-to-CardBus Bridge device Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D3***/***B3*** state but must be lower than the ***D2***/***B2*** state

Present state	Valid next states	PCI-to-CardBus Bridge Requirements in Present State
***D3*$_{cold}$**	Software: • None Hardware: • ***D0*** uninitialized via bus segment reset • Off	Subordinate bus: Off CardBus clock is off CardBus bus is indeterminate except **CRST#** and **CCLK** must be low Interrupts are not available **PME#** is available if *PME_En* is true CardBus card must be in ***D3*** or off Card removal detection is functional Slot Vcc is not changed unless card is removed Ordinate bus: (***B0***, ***B1***, ***B2***) ***B3***: PCI clock off PCI bus is off Interrupts are not available PCI-to-CardBus Bridge responsibilities: Only PME context is preserved if *PME_En* is true Does not respond to any PCI cycles Power consumed by the PCI-to-CardBus bridge is required to be $\leq$ 10 μa unless turning off slot Vcc because a card is removed

©2001 PCMCIA/JEITA

3.5.6 PCI-to-CardBus Bridge Power Management Policies

This section defines the behavior for PCI-to-CardBus bridge functions. The figure below illustrates the areas being discussed in this section.

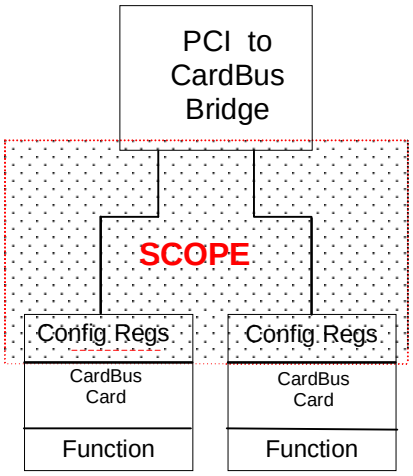


Figure 3-7: Non-Bridge CardBus Function Power Management Diagram

The following tables define the behavior for a PCI-to-CardBus bridge function while operating in each combination of bus and functional power management states.

Legend:

PCI-to-CardBus Bridge Function PM State	Current PCI-to-CardBus bridge function power management state.
PCI-to-CardBus Bridge Bus PM State	Current power management state of the PCI-to-CardBus bridge function's hosting (ordinate) PCI bus segment.
Context	Configuration register and functional state information that are required to be valid for the given power management state. The registers that must remain valid, and the features that must remain available for a given class of device are typically dictated by the corresponding Device-class power management specification. For a PCI-to-CardBus Bridge device, PME context also includes mask and event registers which the operating system will require in order to determine what caused the PME. Also, the bridge controller must not cause the state of the slot Vcc settings to change except when a card is removed in which case the PCI-to-CardBus bridge must turn off slot Vcc.
Power	Power consumption
Access Delay	The minimum required delay before attempting to access the PCI-to-CardBus bridge function to change its power state. If the bus is fully accessible (***B0***) then this delay is solely the result of the state transition delay, or recovery time, following the last write to the function's *PowerState* field. If the bus is not in a fully accessible state (***B1***, ***B2*** or ***B3***) then the delay is characterized by either the function's state transition recovery time, or the time it takes to restore the bus to a fully accessible state, whichever is greater.
Restore Time	The total time from when a PCI-to-CardBus bridge function transitions from its current power management state to the fully configured ***D0*** active state. (Measurement beginning from either a write to the function's **PMCSR**, or a bus segment reset).
Actions to Function	Valid PCI-to-CardBus bridge bus transactions that can be conducted with the PCI-to-CardBus bridge function as the target of the transaction.
Actions from Function	Valid PCI-to-CardBus bridge CardBus and PCI bus transactions, and/or operations that can be initiated by the function.

Table 3-14: *D0* **Power Management Policies**

PCI-to-CardBus Bridge Ordinate (Primary) Bus PM State	PCI-to-CardBus Bridge PM State	PCI-to-CardBus Bridge Context	PCI-to-CardBus Bridge Power	Access Delay	Restore Time	Actions to PCI-to-CardBus Bridge	Actions from PCI-to-CardBus Bridge to CardBus Bus
B0	Legacy PCI-to-CardBus bridge and CardBus card Function ***(D0)***	Full	Full	None	None	Any PCI Transaction	Any CardBus Transaction or Interrupt
B0	***D0*** (Uninitialized)	PME Context*	< as specified by PCI-to-CardBus Bridge vendor	None	None	PCI Configuration Cycles	None (no actions on CardBus bus. CardBus bus is in valid idle state)
B0	***D0*** (Active)	Full	Full	None	None	Any PCI Transaction	Any CardBus Transaction or Interrupt, PME*
B1-B3	***D0*** (Active)	N/A**	N/A**	N/A**	N/A**	N/A**	N/A**

©2001 PCMCIA/JEITA

Table 3-15: *D1* **Power Management Policies**

PCI-to-CardBus Bridge Ordinate (Primary) Bus PM State	PCI-to-CardBus Bridge PM State	PCI-to-CardBus Bridge Context	PCI-to-CardBus Bridge Power	Access Delay	Restore Time	Actions to PCI-to-CardBus Bridge	Actions from PCI-to-CardBus Bridge to CardBus Bus
B0	***D1***	Full Context, PME Context*	≤ ***D0*** uninitialized	None	Need number here	PCI Configuration Cycles	PME only* (CardBus bus is idle)
B1	***D1***	Full Context, PME Context*	≤ ***D0*** uninitialized	Bus restoration time	Need number here	None	PME only* (CardBus bus is idle)
B2-B3	***D1***	N/A**	N/A**	N/A**	N/A**	N/A**	N/A**

Table 3-16: *D2* **Power Management Policies**

PCI-to-CardBus Bridge Ordinate (Primary) Bus PM State	PCI-to-CardBus Bridge Function PM State	PCI-to-CardBus Bridge Context	PCI-to-CardBus Bridge Power	Access Delay	Restore Time	Actions to PCI-to-CardBus Bridge	Actions from PCI-to-CardBus Bridge to CardBus Bus
B0	***D2***	Full Context, PME Context*	< next lower supported PM state, or < ***D0*** uninitialized	200 µs (Note 1)	Need number here	PCI Configuration Cycles	PME only* (CardBus bus is idle, **CCLK** should be stopped)
B1	***D2***	Full Context, PME Context*	< next lower supported PM state, or < ***D0*** uninitialized	Greater of either the bus restoration time or 200 µs (Note 2)	Need number here	none	PME only* (CardBus bus is idle, **CCLK** should be stopped)
B2	***D2***	Full Context, PME Context*	< next lower supported PM state, or < ***D0*** uninitialized	Greater of either the bus restoration time or 200 µs (Note 2)	Need number here	none	PME only* (CardBus bus is idle, **CCLK** should be stopped)
B3	***D2***	N/A**	N/A**	N/A**	N/A**	N/A**	N/A**

Table 3-17: *D3*$_{hot}$ Power Management Policies

PCI-to-CardBus Bridge Ordinate (Primary) Bus State	PCI-to-CardBus Bridge Function PM State	PCI-to-CardBus Bridge Context	Power	Access Delay	Restore Time	Actions to Function	Actions from PCI-to-CardBus Bridge to CardBus Bus
B0	***D3*$_{hot}$**	PCI and functional PME context*	< next lower supported PM state, or < ***D0*** uninitialized	10 ms (Note 1)	Need number here	PCI Config Cycles	PME only* (CardBus bus is indeterminate, **CCLK** is off)
B1	***D3*$_{hot}$**	PME and functional context*	< next lower supported PM state, or < ***D0*** uninitialized	Greater of either the bus restoration time or 10 ms (Note 2)	Need number here	none	PME only* (CardBus bus is indeterminate, **CCLK** is off)
B2	***D3*$_{hot}$**	PME and functional context*	< next lower supported PM state, or < ***D0*** uninitialized	Greater of either the bus restoration time or 10 ms (Note 2)	Need number here	none	PME only* (CardBus bus is indeterminate, **CCLK** is off)
B3	***D3*$_{hot}$**	PME and functional context*	< next lower supported PM state, or < ***D0*** uninitialized	N/A	N/A	none	PME only (CardBus bus is indeterminate, **CCLK** is off)

©2001 PCMCIA/JEITA

Table 3-18: *D3*$_{cold}$ Power Management Policies

PCI-to-CardBus Bridge Ordinate (Primary) Bus State	PCI-to-CardBus Bridge Function PM State	PCI-to-CardBus Bridge Context	Power	Access Delay	Restore Time	Actions to PCI-to-CardBus Bridge	Actions from PCI-to-CardBus Bridge to CardBus Bus
B3	***D3*$_{cold}$**	PCI and Functional PME context only*	No Power from the bus	N/A	full context restore, or boot latency	Bus Segment Reset only	PME only* (CardBus bus is indeterminate)
B3	Legacy PCI Function ***(D3)***	none	No Power	N/A	full context restore, or boot latency	Bus Segment Reset only	none

Notes:

* If PME is supported in this state

** This combination of function and bus power management states is not allowed.

*** Implies device specific, or slot specific power supplies which is outside the scope of this specification.

1. This condition is not typical. It specifies the case where the system software has programmed the function's *PowerState* field and then immediately decides to change its power state again. Typically the state transition recovery time will have expired prior to a power state change request by software.
2. The more typical case where the bus must first be restored to ***B0*** before being able access the function residing on the bus to request a change of its power state. State transition recovery time begins from the time of the last write to the function's *PowerState* field. In this case the bus restoration time is dictated by state transition recovery times incurred in programming the bus's originating device to ***D0*** which then transitions its bus to ***B0***. Bus restoration time is typically the deciding factor in access delay for this case. (see ***3.5.6.1 State Transition Recovery Time Requirements***)

When in ***D1, D2*** or ***D3*$_{hot}$** a PCI-to-CardBus bridge must not respond to PCI transactions targeting its I/O or memory spaces or assert a functional interrupt request.

A PCI-to-CardBus bridge cannot tell the state of its PCI bus; therefore, it must always be ready to accept a PCI configuration access when in ***D1***, ***D2*** or ***D3*$_{hot}$**.

3.5.6.1 State Transition Recovery Time Requirements

The following table shows the minimum recovery times (delays) that must be guaranteed, by hardware in some cases and by system software in others, between the time that a function is programmed to change state and the time that the function is next accessed (including PCI configuration space).

Table 3-19: PCI-to-CardBus Bridge State Transition Delays

Initial State	Next State	Minimum System Software Guaranteed Delays
D0	***D1***	0
D0 or ***D1***	***D2***	200 µs
D0, ***D1*** or ***D2***	$D3_{hot}$	10 ms
D1	***D0***	0
D2	***D0***	200 µs
$D3_{hot}$	***D0***	10 ms

3.6 PCI-to-CardBus Bridges and Power Management

With power management under the direction of the operating system each class of devices (PCI and CardBus functions) must have a clearly defined criteria for feature availability as well as what functional context must be preserved when operating in each of the power management states. Some example Device-Class specifications have been proposed as part of the Intel / Microsoft / Toshiba ACPI specification for various functions ranging from audio to network adapters. While defining Device-Class specific behavioral policies for most functions is well outside of this specification's scope, defining the required behavior for the PCI-to-CardBus bridge functions is within the scope of this specification. The definitions here apply to the PCI-to-CardBus Bridge.

The mechanisms for controlling the state of these function vary somewhat depending on which type of originating device is present. The following sections describe how these mechanisms work for the PCI-to-CardBus bridge.

This section details the power management policies for PCI-to-CardBus Bridge functions. The PCI-to-CardBus bridge function can be characterized as an originating device with a secondary bus downstream of it. The relationship of the bridge function's power management state to that of its secondary bus has been mentioned in ***3.4.7.1 Control of Secondary Bus Power Source and Clock*** and will be developed further in this section.

The shaded regions in the figure below illustrate what will be discussed in this section.

©2001 PCMCIA/JEITA

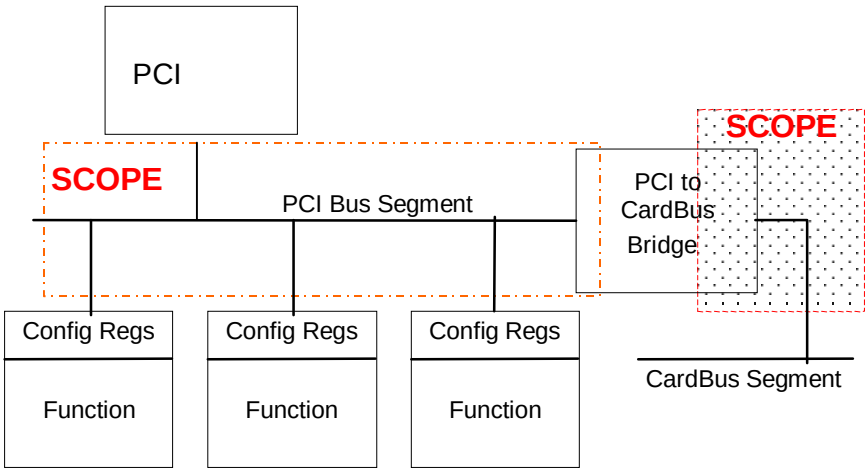


Figure 3-8: PCI Bridge Power Management Diagram

As can be seen above the PCI-to-CardBus bridge behavior to be described in this section is common, from the perspective of the operating system, to both host bridges, and PCI-to-CardBus bridges.

The following table defines the relationship between a bridge function's power management state, and that of its secondary bus. Also detailed are the resultant attributes of the secondary bus. The rightmost column of the table details a set of conditions that all "downstream" PCI functions must be capable of withstanding when residing on a bus in a given state without their application breaking in a way that cannot be gracefully recovered from.

It is the responsibility of the system software to ensure that only valid, workable combinations of bus and downstream PCI and CardBus bus function power management states are used for a given PCI and CardBus bus and all PCI and CardBus functions residing on that bus.

Legend:

Bridge PM State	Current PCI power management state of bridge.
Secondary PCI or CardBus Bus State	Current power management state of the originating device's (bridge function's) PCI bus segment.
Secondary CardBus Attributes	The characteristics of the secondary bus for the current bus power management state.
Downstream Function Attributes	Necessary attributes of a PCI or CardBus function residing on the secondary bus given the secondary bus's power management state.

Table 3-20: PCI-to-CardBus Bridge Power Management Policies

Present state	Valid next states	PCI-to-CardBus Bridge Requirements in Present State
D0 (unitialized)	Software: • ***D0*** (active) • ***D3*** Hardware: • ***D0*** uninitialized via PCI bus segment reset • Off	Subordinate bus: Vcc: Off; no change in applied Vcc if coming from D3cold under control of PCI-to-CardBus Bridge V_{AUX} and *PME_En* is true unless card is removed Bus: Per ***PC Card Standard*** for appropriate card technology Ordinate bus: ***B0*** PCI clock is running unless clock run is asserted. If clock run is not functional, clock is running. PCI bus is fully functional PCI (and ISA if implemented) interrupts are not functional PCI-to-CardBus Bridge responsibilities: Context (PCI bus in ***B0***) *PME_En* false Card detect pins valid, card type valid, slot Vcc off, context as programmed by BIOS or power on defaults *PME_En* true PME context; no change in Vcc to slot unless card is removed Bridge responds appropriately to configuration cycles for this PCI-to-CardBus Bridge device Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D0*** uninitialized state

©2001 PCMCIA/JEITA

Present state	Valid next states	PCI-to-CardBus Bridge Requirements in Present State
D0 (active)	Software: • ***D1*** • ***D2*** • ***D3*** Hardware: • ***D0*** *uninitialized via PCI bus segment reset* • *Off*	Subordinate bus: ***B0*** CardBus clock is running unless Clock Run is asserted. If clock run is not functional in the PCI-to-CardBus bridge, clock is on CardBus bus is fully functional CardBus card functional and CardBusStatusChange interrupts are functional PCI-to-CardBus Bridge, CardBus and 16-bit PC Card **PME#** is available if *PME_En* is true CardBus card must be in ***D0***, ***D1***, ***D2*** or ***D3*** Slot Vcc is not changed unless card is removed Ordinate bus: ***B0*** PCI clock is running unless clock run is asserted. If clock run is not functional, clock is running. PCI bus is fully functional PCI (and ISA if implemented) interrupts are functional PCI-to-CardBus Bridge responsibilities: All context, PCI and functional, are available. Bridge responds appropriately to all PCI cycles for this PCI-to-CardBus Bridge device, its 16-bit PC Card and CardBus cards Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D0/B0*** state
D1	Software: • ***D0*** • ***D2*** • ***D3*** Hardware: • ***D0*** uninitialized via PCI bus segment reset • Off	Subordinate bus: ***B1*** CardBus clock is running unless Clock Run is asserted. If clock run is not functional in the PCI-to-CardBus bridge, clock is on CardBus bus is idle CardBus card functional and CardBusStatusChange interrupts are not available PCI-to-CardBus Bridge and CardBus **PME#** is available if **PME_En** is true CardBus card must be in ***D1***, ***D2*** or ***D3*** Card removal detection is functional Slot Vcc is not changed unless card is removed Ordinate bus: (***B0***) ***B1***: PCI clock is running unless clock run is asserted. If clock run is not functional, clock is running. PCI bus may be idle PCI (and ISA if implemented) interrupts are not available PCI-to-CardBus Bridge responsibilities: All context, PCI and functional, is preserved; PCI-to-CardBus Bridge PCI configuration space is functional; functional context is not available Bridge only responds to PCI type 0 cycles for this PCI-to-CardBus Bridge device Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D1/B1*** state but must be lower than the ***D0/B0*** state

Present state	*Valid next states*	PCI-to-CardBus Bridge Requirements in Present State
D2	Software: • ***D0*** • ***D3*** Hardware: • ***D0*** *uninitialized via PCI bus segment reset* • Off	Subordinate bus: ***B2*** CardBus clock is not running unless Clock Run is required by CardBus card. If clock run is not functional in the PCI-to-CardBus bridge, clock is off CardBus bus is idle CardBus card functional and CardBusStatusChange interrupts are not available PCI-to-CardBus Bridge and CardBus **PME#** is available if *PME_En* is true CardBus card must be in ***D2*** or ***D3*** Card removal detection is functional Slot Vcc is not changed unless card is removed Ordinate bus: (***B0***, ***B1***) ***B2***: PCI clock is not running unless clock run is required. If clock run is not functional, clock is off. PCI bus is idle PCI (and ISA if implemented) interrupts are not available PCI-to-CardBus Bridge responsibilities: All context, PCI and functional, is preserved; PCI-to-CardBus Bridge PCI configuration space is functional; functional context is not available Bridge only responds to PCI type 0 cycles for this PCI-to-CardBus Bridge device Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D2*** / ***B2*** state but must be lower than the ***D1*** / ***B1*** state

Present state	*Valid next states*	PCI-to-CardBus Bridge Requirements in Present State
***D3*$_{hot}$**	Software: • ***D0*** uninitialized Hardware: • ***D0*** uninitialized via PCI bus segment reset • ***D3*$_{cold}$** if V_{AUX} supplied • Off	Subordinate bus: ***B3*** CardBus clock is off CardBus bus is indeterminate except **CRST#** and **CCLK** are low Interrupts are not available PCI-to-CardBus Bridge and CardBus **PME#** is available if *PME_En* is true CardBus card must be in ***D3*** Card removal detection is functional Slot Vcc is not changed unless card is removed Ordinate bus: (***B0***, ***B1***, ***B2***) ***B3***: PCI clock off PCI bus is off Interrupts are not available PCI-to-CardBus Bridge responsibilities: Only PME context is preserved if *PME_En* is true; PCI-to-CardBus Bridge PCI configuration space is readable; functional PME context is not available; only PCI-to-CardBus Bridge Power Management Control and Status Register *PowerState*, *PME_En* and *PME_Status* bits are required to be functional Bridge only responds to PCI type 0 cycles for this PCI-to-CardBus Bridge device Power consumed by the PCI-to-CardBus bridge is as specified by the vendor for the ***D3***/***B3*** state but must be lower than the ***D2***/***B2*** state

©2001 PCMCIA/JEITA

Present state	*Valid next states*	PCI-to-CardBus Bridge Requirements in Present State
***D3*$_{cold}$**	Software: • None Hardware: • ***D0*** uninitialized via PCI bus segment reset • Off	Subordinate bus: Off CardBus clock is off CardBus bus is indeterminate except **CRST#** and **CCLK** are low Interrupts are not available **PME#** is available if *PME_En* is true CardBus card must be in ***D3*** or off Card removal detection is functional Slot Vcc is not changed unless card is removed Ordinate bus: (***B0***, ***B1***, ***B2***) ***B3***: PCI clock off PCI bus is off Interrupts are not available PCI-to-CardBus Bridge responsibilities: Only PME context is preserved if *PME_En* is true Does not respond to any PCI cycles Power consumed by the PCI-to-CardBus bridge is required to be $\leq$ 10 μA unless turning off slot Vcc because a card is removed

3.6.1 PCI-to-CardBus Bridge

While a PCI-to-CardBus Bridge behaves much the same as a PCI to PCI bridge from a functional point-of-view, The CardBus specification already defines its own power management capabilities. These capabilities should be used directly to control the power management state of each 16-bit PC Card socket. CardBus devices wishing to support common silicon with PCI, however should implement the registers defined in ***3.3 PCI-to-CardBus Bridge Power Management Interface*** of this specification. CardBus bridges should implement the power management registers in their configuration space to allow the bridge silicon and its primary bus to be power managed.

CardBus does not define a **PME#** signal, however it does define a signal with similar intended usage. The Card Status Change signal will be used as a source for power management events for a CardBus card and Status Change will be used for a PC Card. The PCI-to-CardBus Bridge will use this event as one condition for assertion of **PME#**. Note that **PME#** must adhere to this specification (see ***3.7 Power Management Events***).

One other minor difference in this specification for a PCI-to-CardBus bridge is the location of the New Capabilities Pointer in the PCI Config Space Header. For CardBus bridges the **Cap_Ptr** register is found at location 14H of the PCI configuration space header.

Unlike other PCI bridges, the PCI-to-CardBus bridge comes with its own set of power management events. These events, and hence PME context, are the ExCA Card Status-Change Interrupt Configuration Register and the ExCA Card Status-Change Register. In ExCA space, the ExCA Card Status-Change Interrupt Configuration Register PME bits are defined to be at index 805H and are:

Table 3-21: PME Context: ExCA Card Status-Change Interrupt Configuration Register, 805H

Name	Description
Card detect change	Enables status change interrupts when a card is inserted or removed
Ready change	Enables, for a 16-bit memory card, when the **READY** signal transitions from BUSY to READY
Battery warning change	Enables, for a 16-bit memory card, battery warning conditions
Status change	Enables battery dead condition or a card asserting status change (**STSCHG/RI#**) for a 16-bit PC Card.

Likewise, PME context for a PCI-to-CardBus bridge controller must include the ExCA Card Status-Change Register (ExCA index 804H). This context which must be preserved according to the rules of PME context preservation are:

Table 3-22: PME Context: ExCA Card Status-Change Register, 804h

Signal	Description
Card detect changed	A change was detected on either **CD1#** or **CD2#**
Ready	A change was detected due to a Busy to Ready transition
Battery warning	A battery warning condition exists
Battery dead Status change Ring Indicate	A battery dead (**BVD**) condition exists for a 16-bit memory PC Card, or a status change (**STSCHG#**) condition exists for a PC Card, or a card status change (**CSTSCHG**) condition exists for a CardBus card, or a ring indicate (**RI#**) condition exists

The PCI-to-CardBus Bridge PME context for the PCI-to-CardBus Bridge for CardBus sockets is defined as:

Table 3-23: PCI-to-CardBus Bridge PME Context for Bridge Devices

Socket Event Register, offset CardBus Socket Address + 00H, Bit	Description
3	PowerCycle
2	CardBusCardDetect2#
1	CardBusCardDetect1#
0	CSTSCHG
Socket Mask Register, offset CardBus Socket Address + 04H, Bit	**Description**
3	PowerCycle mask
2::1	CardDetect mask
0	CSTSCHG mask

©2001 PCMCIA/JEITA

The Vcc settings must be preserved by the following registers:

ExCA Power Control Register index 802H	CardBus Socket Control Register CardBus Socket Address + 10H	
16-bit PC Card Vcc Control Register	**Bits 6::4**	**Description**
	000	Socket off
	001	reserved
	010	5V
	011	3.3V
	100	X.XV
	101	Y.YV
	110	Reserved
	111	Reserved

These bits reflect the current state of the card's signal except when the controller is in the ***D3*** state. When the controller is transitioned to the ***D0*** state it should reflect the current signal condition.

Finally, PME context for the PCI-to-CardBus bridge must include the *PME_Support* bit, bit 15, for $\boldsymbol{D3_{cold}}$ in the Power Management Control and Status Register.

A pin must be dedicated on the PCI-to-CardBus bridge controller to perform the **PME#** function. **PME#** is asserted when any of the conditions are present as indicated by the ExCA Card Status-Change Registers or the CardBus Socket Event Register and enabled by programming the ExCA Status-Change Interrupt Configuration Register or the CardBus Socket Mask Register. When PME is enabled, *PME_En* is set, interrupts normally associated with controller status change interrupts shall be routed to the **PME#** pin instead of the controller status change interrupt. CardBus card functional and status change interrupts are not functional except in the ***D0*** state.

Finally, when power management events are enabled, *PME_En* is set, and the PCI-to-CardBus bridge controller is transitioned from ***D3*** to ***D0*** by programming the power state bits, the PCI-to-CardBus bridge controller must not cause the slot Vcc setting to change. Likewise, when power management events are enabled and *PME_En* is set, neither can the PCI-to-CardBus bridge controller cause the slot Vcc setting to change as a result of a bus segment reset when in the ***D3*** state or when programmed to enter the ***D3*** state. Slot Vcc states must initialize to "off" when the PCI-to-CardBus bridge controller is powered up in a ***D0*** uninitialized state. The operating system is responsible for all other Vcc changes. However, when the PCI-to-CardBus Bridge is powered, either by V_{AUX} or Vcc, and a card is removed, the PCI-to-CardBus Bridge must turn off Vcc to the appropriate slot to ensure a card is inserted into a cold socket.

In addition to supporting legacy card status change interrupts, the **STSCHG#**/**CSTSCHG** function also performs the **PME#** function. This implies there are two drivers involved and, in fact, there are. In an ACPI architecture, there exists a device class driver and a bus class driver. The device class driver is today's legacy driver. If the device class driver enables the ExCA Card Status-Change Interrupt Configuration mask register, index 805H, then the status change interrupt must be routed out the programmed interrupt in the same register; likewise, if the device class driver enables the CardBus Socket Mask register, then the appropriate status change interrupt occurs. If, on the other hand, **PME#** is enabled from the bus class driver, the status change interrupt for either card must be routed out the PCI-to-CardBus Bridge controller's **PME#** pin. If the status change interrupt occurs, it is reflected in the *PME_Status* bit in the Power Management Control/Status – **PMCSR** register. See ***3.3.2.4 Power Management Control/Status - PMCSR (Offset = 4)***.

3.7 Power Management Events

The Power Management Event (**PME#**) signal is an open drain, active low signal that is to be driven low by a PCI function to request a change in its current power management state and / or to indicate that a power management event has occurred.

The assertion and deassertion of **PME#** is asynchronous to the PCI clock.

In general, this signal is bussed between all PCI connectors and devices in a system although certain implementations may choose to pass separate buffered copies of the signal to the system logic. Devices must be enabled by software before asserting this signal. Once asserted, the device must continue to drive the signal low until software explicitly clears the *PME_Status* or *PME_En* bit in the **PMSCR** register of the function. The system vendor must provide a pull-up on this signal if it allows the signal to be used. Systems vendors that do not use this signal are not required to bus it between connectors or provide pull-ups on those pins. *PME_Status* and *PME_En* are only cleared when written with a "1".

PME# is not intended to be used as an input by any PCI function, and the system is not required to route or buffer it in such a way that a function is guaranteed to detect that the signal has been asserted by another function.

Software will enable its use by setting the *PME_En* bit in the **PMCSR**. When a PCI function generates or detects an event which requires the system to change its power state (e.g. the phone rings), the function will assert **PME#**. It must continue to assert **PME#** until software either clears the *PME_En* bit or clears the *PME_Status* bit in the **PMCSR** even if the source of the power management event is no longer valid (e.g. the phone stops ringing) . Some devices which are powered by a battery or some external power source may use this signal even when powered off. Such devices must maintain the value of the *PME_Status* bit through reset.

PME# has additional electrical requirements over and above standard open drain signals that allow it to be shared between devices which are powered off and those which are powered on. The additional requirements include careful circuit design to ensure that a voltage applied to the **PME#** network will never cause damage to a component even if that particular component's Vcc inputs are not powered. Additionally the device must ensure that it does not pull **PME#** low unless **PME#** is being intentionally asserted in all cases including when the function is in $\boldsymbol{D3_{cold}}$.

What this means is that any component implementing **PME#** must be designed such that:

1. Unpowered **PME#** driver output circuits will not be damaged in the event that a voltage is applied to them from other powered "wire ORed" sources of **PME#** .
2. When power is removed from its **PME#** generation logic, the unpowered output does NOT present a low impedance path to ground or any other voltage.

These additional requirements ensure that the **PME#** signal network will continue to function properly when a mixture of powered, and unpowered components have their **PME#** outputs wire or'd together. It is extremely important to note that most commonly available open drain, and tri-state buffer circuit designs used "as is" do NOT satisfy the additional circuit design requirements for **PME#**.

Other requirements on the motherboard / add-in card designer include:

1. Common ground plane across the entire system
2. Split voltage power planes (V_{AUX} vs. Vcc) are allowed
3. Vcc at the socket connector must be switched off by the operating system except when a card is removed.

©2001 PCMCIA/JEITA

The card designer must be aware of the special requirements that constrain **PME#** and ensure that their card does not interfere with the proper operation of the **PME#** network. **PME#** input into the system may deassert as late as 100ns after the **PME#** output from the function deasserts.

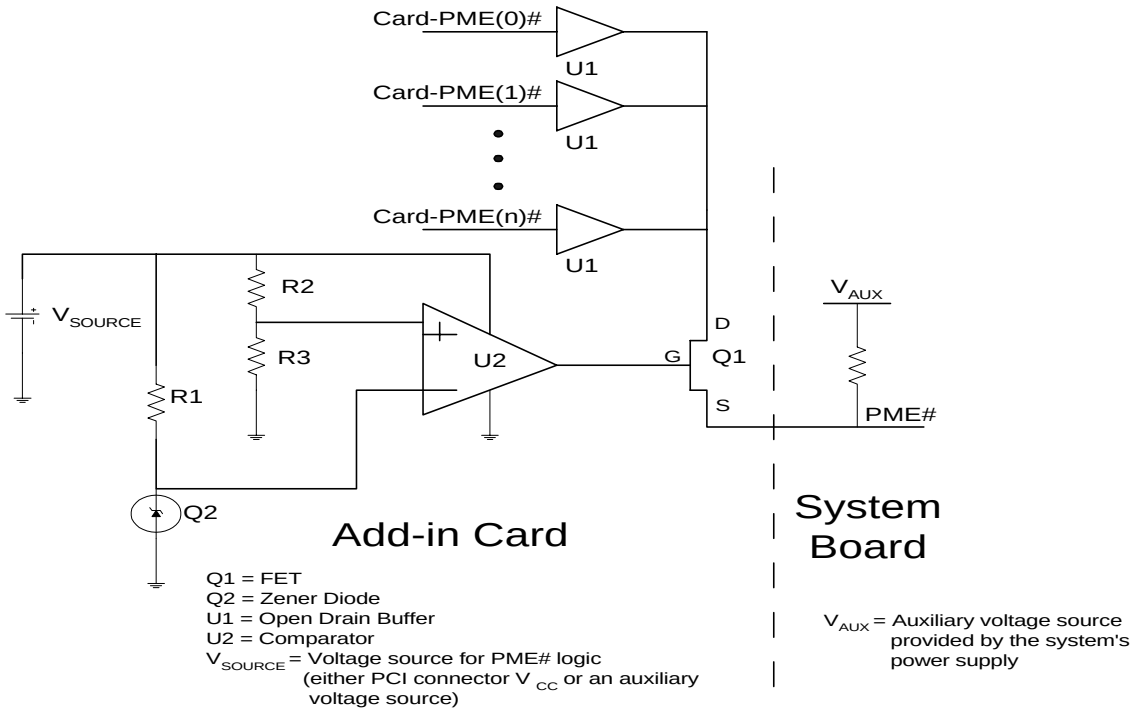
The value of the pull-up resistor for **PME#** on the System Board should be derived taking into account the output capacitance of the open drain driver to ensures that **PME#** charges up to a logic high voltage level in no greater than 100 ns. (See Section 4.3.3 of the ***PCI Local Bus Specification, Revision 2.1*** for information on pullup resistors.)

Implementation Note: Example PME# **Circuit Design**

The following diagram is an example of how the **PME#** generation logic could be implemented. In this example multiple PCI functions, perhaps Wire "OR'd" around a PCI to PCI bridge on the add-in card, have their ganged output connected to the single **PME#** pin on the PCI slot connector.

The circuit driving the gate of transistor Q1 is designed to isolate the card's **PME#** network from that of the System Board whenever its power source (V_{SOURCE}) is absent.

If the card supplies power to its **PME#** logic with the PCI connector's Vcc (i.e. it does not support **PME#** from $D3_{cold}$), then all of the **PME#** sources from the card will be isolated from the system board when the slot's VCC is switched off. Cards that support **PME#** from $D3_{cold}$ have an auxiliary power source to power the **PME#** logic which will maintain connection of these **PME#** sources to the System Board network even when the bus power (Vcc) has been switched off.



This example assumes that ALL sources of **PME#** on the add-in card are powered by either Vcc or V_{AUX} (V_{SOURCE}). If **PME#** from $D3_{cold}$ is supported by some, but not all of the card's functions that generate **PME#**, then the card designer must ensure that there is separate isolation control for each of the **PME#** generation power sources.

PCI component designers could choose to integrate the "power fail detect" isolation circuitry with their **PME#** output pin physically corresponding to the source of FET Q1. Alternatively all isolation control logic could be implemented externally on the add-in card.

Note: This example is meant as a conceptual aid only, and is not intended to prescribe an actual implementation.

3.7.1 Power Management Event (PME#) Routing

PME# is routed throughout the system in the same way that PCI Bus functional interrupt requests are routed. All sources of **PME#** are typically connected together (wire "OR"ed) to present a single point of connection into the system.

In the case of the PCI-to-CardBus bridge, **PME#** routing goes through the bridge. This is because of the level and sense transitions required for CardBus cards and 16-bit PC Cards and because the PCI-to-CardBus bridge itself is capable of generating Power Management Events. The PCI-to-CardBus bridge generates **PME#** when *PME_En* is set rather than status change interrupts. This is shown below in Figure 3-9: **PME#** System Routing.

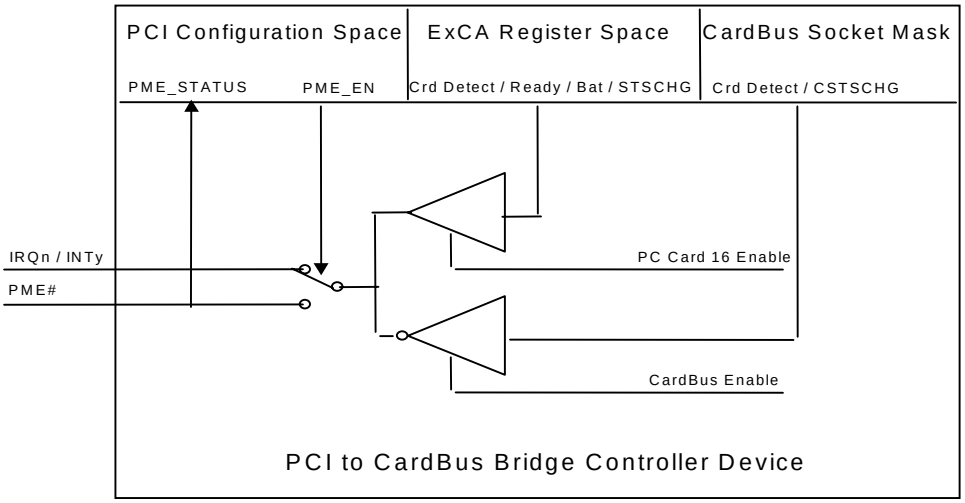


Figure 3-9: PME# **System Routing**

System software, having built itself a table of PCI functions that have been enabled for **PME#** generation, will walk this table at the start of the PME service routine to determine which PCI function, or functions have requested a change in their power management state or for which a power management event has occurred.

Large system designs, with correspondingly large PCI bus hierarchies, might consider a system specific way of presenting multiple parallel **PME#** subnet connections into the system as a means of potentially reducing the time it takes for system software to isolate which PCI function, or functions have requested PME service. However these implementations are beyond the scope of this specification.

Systems must route the **PME#** signal to the appropriate system logic to wake the system. For example, an Intel / Microsoft / Toshiba ACPI compliant systems may route this signal to the **SCI#** interrupt. Systems which support waking up from a "soft off" or a low power suspend where significant portions of the system are powered off may route the signal to a power sequencing state machine.

©2001 PCMCIA/JEITA

3.7.2 Auxiliary Power for *D3*$_{cold}$ Power Management Events

Power managed systems that support PME generation while in the ***D3*$_{cold}$** state may require an auxiliary power source. There are several ways to provide auxiliary power for any necessary "keep alive" circuitry including but not limited to:

1. On-Board Battery
2. AC "Brick" adapter, externally provided power source
3. Auxiliary power supplied by the system

In the case of the PCI-to-CardBus bridge controller, the auxiliary supply provided for the controller will be limited to 10 µa. The controller must not require more than 10 µa when:

1. *PME_En* is set true in the PCI-to-CardBus bridge's **PMCSR**
2. The PCI-to-CardBus bridge controller is programmed to the ***D3*** state
3. The originating (ordinate) PCI clock is stopped
4. The originating (ordinate) PCI bus is idle

The transition from Vcc to V_{AUX} can occur only after the above three conditions are implemented. Additionally, the transition from Vcc to V_{AUX} must include the stable application of V_{AUX} for a minimum of 50 ms prior to transitioning Vcc off. Likewise, when transitioning from off to on, V_{AUX} must remain stable for 50 ms after Vcc is up and stable. See ***Figure 3-10: Vcc to VAUX Transitioning*** for more information.

Since neither a CardBus nor a PC Card has no provisions or capabilities for adding a V_{AUX} pin, Vcc will provide the V_{AUX} function. The operating system and system control methods will be required to determine if the host platform can support the ***D3*$_{cold}$** current requirements of the card before performing the power management functions.

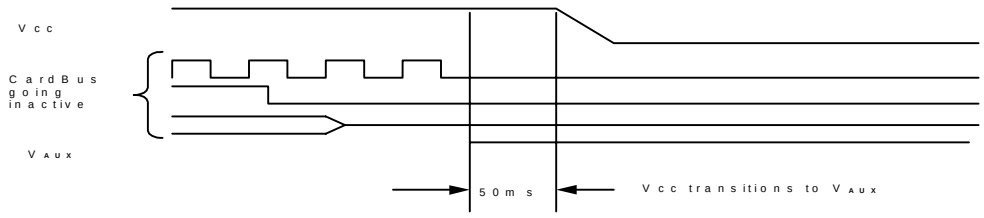


Figure 3-10: Vcc to V_{AUX} Transitioning

3.8 Software Support for PCI Power Management

The PCI Power Management specification defines the requirements for all PCI functions to be managed by an Operating System. The specification does not attempt to define any sort of Power Management Policy. That is left up to the individual Operating System. However, it is necessary to address some of the basic assumptions that have been made regarding which aspects of power management are enforced by system software versus by hardware such that the functions might react appropriately. These assumptions fall into four basic categories: Identifying PCI Function Capabilities, Placing PCI Functions in a Low Power State, Restoring PCI Functions from a Low Power State, and Wake Events. Within each of these areas, it has been assumed that the Operating System

software will handle certain power management functions in addition to it's basic power management policy. It must be noted that in the context of this document references to the Operating System software include any device drivers and other Operating System specific power management services.

3.8.1 Identifying PCI and CardBus Function Capabilities

The Operating System software is responsible for identifying all PCI function power capabilities by traversing the New Capabilities structure in PCI Configuration space, and reading the Power Management Capabilities Register (**PMC**). It is the Operating System's responsibility to ensure that it does not attempt to place a function into a power management state which the function does not support. If, for any reason, the Operating System software attempts to put a function into a power management state that the function does not support, the function should handle this gracefully[5], and remain in whatever state it was in before the request. The *PowerState* bits of the **PMCSR** will reflect the current state of the function, not the intended invalid state which was written.

3.8.2 Placing PCI and CardBus Functions in a Low Power State

When attempting to place a PCI function in a low power state ***D1 - D3***, it is the Operating System's responsibility to ensure that the function has no pending (host initiated) transactions, or in the case of a Bridge device, that there are no PCI functions behind the bridge that require the bridge to be in the fully operational ***D0*** state. Furthermore, it is the Operating System's responsibility to notify any and all device drivers that are conducting peer-to-peer transfers to the target function that the target function will no longer be accessible. In other words, it is the Operating System's responsibility to ensure that no peer-to-peer activity occurs with the sleeping PCI function as the target. If the Operating System and the PCI function both support Wake Events, the Operating System should enable the function's Power Management Event (**PME#**) line via the *PME_En* bit in the function's Power Management Control Register (**PMCSR**).

3.8.2.1 Buses

When attempting to place a PCI Bus segment into a lower power management state (***B1- B3***) the Operating System must first ensure that all PCI functions on that bus have been placed in an appropriate low power state. The same is true for CardBus buses and cards.

3.8.2.2 *D3* State

Prior to placing a PCI-to-CardBus bridge function into ***D3***, the Operating System must determine if it has the capability to restore the function from this state. Restoration includes reinitializing the function. The Operating System must make sure that it has the appropriate driver loaded for that PCI-to-CardBus bridge function in order to restore the functions to operation. If the Operating System is not capable of fully reinitializing a device, then the Operating System should not put the device into ***D3***. When placing a function into ***D3***, the Operating System Software is required to disable I/O and memory space as well as Bus Mastering via the PCI Command Register.

[5] Finish the PCI transaction with normal completion and ignore the write data. This is NOT a hardware error condition.

©2001 PCMCIA/JEITA

3.8.3 Restoring PCI Functions From a Low Power State

3.8.3.1 *Dx* States and the DSI Bit

For support of the DSI (Device Specific Initialization) bit, consult the PCI Bus Power Management Specification.

3.8.4 Wake Events

3.8.4.1 Wake Event Support

PCI and CardBus Power Management supports Wake events generated by functions on the PCI bus. Assuming the Operating System supports Wake Events, it is the system hardware's responsibility to restore the Host processor subsystem to a state which will permit the Operating System to function (through ACPI or some other architecture). If the sub-system is already in a ***D0*** state, then the system hardware does not need to take any special action. The System is responsible for notifying the Operating System that a PCI or CardBus Power Management Event has occurred. It is expected that **PME#** will generate some form of System Control Interrupt (SCI), but whether this interrupt is handled by a device driver or an Operating System service routine, is left up to the individual Operating System architecture.

Once the Operating System has been notified that a PCI PME has occurred, it is the Operating System's responsibility to restore power to the primary PCI or CardBus bus and to restore it to the ***B0*** state, then to restore power to any unpowered slots / devices, and finally query the PCI or CardBus functions that have been configured with **PME#** enabled to determine which function, or functions had generated **PME#.** If the generating device is a bridge device, the Operating System should follow this procedure for any subsequent PCI bridges. Should **PME#** become deasserted before the Operating System identifies the device which generated it, or before the **PME#** is serviced, the Operating System must recover gracefully. Furthermore, the Operating System must be able to handle multiple **PME#**s generated by different functions simultaneously. Upon identifying the source or sources of the **PME#,** it is up to the Operating System Power Management Policy to identify the correct course of action with regard to waking the functions and / or the rest of the system.

3.8.4.2 The *D0* "Initialized" State From a Wake Event

Before the Operating System returns a function to ***D0*** which will require a re-initialization of the function, it must ensure that the Operating System not only has the information necessary to re-initialize the function, but also any information necessary to *restore the function* as well. This information is often client specific. As an example, assume a modem's client has set up a modem function in a specific state additional to default initialization (error correction, baud rate, modulation characteristics, etc.). The client / function then goes unused for an extended amount of time *which may cause* the power manager to place the modem in a ***D2*** or perhaps even a ***D3*** state.

When the client is called upon to interact with the modem (such as a ring-resume event), the Operating System will have transitioned the modem function to the ***D0*** initialized state. However restoration of the modem function to ***D0*** alone may not be sufficient for the function and client to perform the indicated task. It is manifest upon Device-Class specifications to include sufficient context save requirements for successful restoration of a function. The restoration must be transparent to the extent that the host application is unaware that a power state transition and the associated restoration occurred.

Transitioning from ***D0 unitialized*** to ***D0 active*** is defined when the appropriate memory, I/O windows and/or busmastering enable bits are enabled by the operating system.

3.8.4.3 Device Class Specification for PCI-to-CardBus Bridges

Following is the Device Class Specification for PCI-to-CardBus bridge devices:

D0

- Card status change (from slot): Functional
- Functional interrupts to processor: Functional
- Controller context (e.g. memory, I/O windows): Fully functional
- Controller interface: Fully functional (processor can access cards)
- Power management block in configuration space: Fully accessible
- Power to cards (slots): Available
- Power level consumption by controller: Highest power consumption. This is limited by the PCI Local Bus Specification
- Allowable power management state changes for controller: ***D1***, ***D2*** and ***D3***
- Restore time: Not applicable
- Bus command response time: Fastest
- Expected PC Card power state: Any of the following: ***D0***, ***D1***, ***D2***, and ***D3***
- In ***D0*** state, **CSTSCHG** interrupts can be passed to a system from a powered down PC card (for more detail, refer to ***System and Interface Wakeup*** of the ***Electrical Specification***).

D1

- Card status change (from slot): Functions as **PME#** if *PME_EN* is enabled if *PME_EN* is true
- Card functional interrupts: Not functional
- PME Context: functional
- Controller context (e.g. memory, I/O windows BARs): Preserved but can not be accessed except in ***D0*** state
- Controller interface: Non-functional (must be in state ***D0***)
- Slot clock: On unless Clock Run protocol has it stopped
- Slot bus: Idle
- PCI Configuration Space: PCI Configuration Space preserved, Power management block in configuration space can be accessed but only PowerState bits are required to be writeable
- Power to cards (slots): Available
- Power level consumption for controller: High but less than ***D0***.
- Allowable power management state changes for controller: ***D0***, ***D2*** and ***D3***

©2001 PCMCIA/JEITA

- Restore time: The time required to restore the function from the ***D1*** state to the ***D0*** state is quicker than resumption from ***D3***
- Bus command response time: Equal to or slower than in ***D0***
- Expected CardBus Card state: Any of the following: ***D1***, ***D2***, and ***D3***
- In ***D1*** state, **CSTSCHG** interrupts can be passed to a system from a powered down PC card (for more detail, refer to ***System and Interface Wakeup*** of the ***Electrical Specification***).
- **CSTSCHG** interrupt events cause the Power Management Event pin to be asserted if wakeup is enabled (*PME_EN* true).

D2

- Card status change (from slot): Functions as **PME#** if *PME_EN* is enabled if *PME_EN* is true
- PME Context: functional
- Controller context (e.g. memory, I/O windows BARs): Preserved but can not be accessed except in ***D0*** state
- Card functional interrupts: Not functional
- Controller interface: Non-functional (must be in state ***D0***)
- Slot clock: May be on but should be stopped (Clock Run protocol)
- Slot bus: Idle
- PCI Configuration Space: PCI Configuration Space preserved, Power management block in configuration space can be accessed but only PowerState bits are required to be writeable
- Power to cards (slots): Available
- Power level consumption for controller: Less than ***D1***.
- Allowable power management state changes for controller: ***D0*** and ***D3***
- Restore time: The time required to restore the function from the ***D2*** state to the ***D0*** state is quicker than resumption from ***D3***
- Bus command response time: Equal to or slower than in ***D1***
- Expected CardBus Card state: Any of the following: ***D2*** and ***D3***
- Clock (**CCLK** to socket) is stopped

D3 (*PME_En* not enabled)

- Card status change (from slot): Not detectable, **PME#** not available
- Controller context (e.g. memory, I/O windows): Lost
- Controller interface: Non-functional (processor can not access cards)
- PCI Configuration Space: PCI Configuration Space preserved, Power management block in configuration space can be accessed but only PowerState bits are required to be writeable
- Card functional interrupts to processor: Non-functional
- Clock to controller: May be off

- Socket interface: indeterminate

D3 (V_{AUX} supplied, *PME_En* enabled)

- Card status change (from slot): Functions as **PME#** if *PME_EN* is enabled if *PME_En* is true
- PCI and function PME Context: functional, all other context lost
- Controller interface: Non-functional (processor can not access cards)
- Card functional interrupts to processor: Non-functional
- Upon a card removal, the PCI-to-CardBus Bridge is required to turn off slot power and new state (off) is reflected in Vcc control register as appropriate
- Clock to controller: Off
- Socket interface: indeterminate

If Vcc is removed (e.g. PCI Bus ***B3***) while the device is in the ***D3*** state, a bus-specific reset (e.g. PCI **RST#**) must be asserted when power is restored and functions will then return to the ***D0*** state with a full power-on reset sequence. Whenever the transition from ***D3*** to ***D0*** is initiated through assertion of a bus-specific reset, the power-on defaults will be restored to the function by hardware just as at initial power up except for PME context when *PME_En* is enabled. The function must then be fully initialized and configured by software.

3.8.4.4 PME Context for PCI-to-CardBus Bridges

As stated earlier, the PCI-to-CardBus bridge is unusual in that it is the only bridge to date that has PME functional context. The PME functional context is summarized as follows:

- *PME_En*
- *PME_Status*
- *PME_Support* bit 15 – $\boldsymbol{D3_{cold}}$
- ExCA Status-Change Register(s) (index 804H)
  - Card detect
  - Ready
  - Battery Warning
  - Battery dead or Status Change
- ExCA Status-Change Interrupt Configuration Register(s) (index 805H)
  - Card detect enable
  - Ready enable
  - Battery warning enable
  - Battery dead / status change enable
- CardBus Socket Mask Register(s) (CardBus socket address + 04H)
  - Card detect
  - Card status change

©2001 PCMCIA/JEITA

- CardBus Socket Event Register(s) (CardBus socket address + 00H)
  - Card detect pins
  - Card status change
- Vcc settings for slots
  - ExCA Power Control Register, 802H (and any other device specific as required)
  - CardBus Socket Control Register, Socket Address + 10H, bits 6::4

The type of card installed determines what type of PME context is to be preserved: if a 16-bit PC Card is present, then the ExCA register set is the correct PME context; if a CardBus card is present, then the CardBus Socket register set is the correct PME context. *PME_EN* and *PME_STATUS* is always preserved.

For the PCI-to-CardBus bridge, function device context (context saved by the device class driver) is defined as follows:

- type(s) of cards installed (CardBus or 16-bit PC Card)
- base address register values

3.8.5 Get Capabilities

The Get Capabilities operation is performed by the operating system as it is enumerating devices in the system. Get Capabilities tells the Operating System information about what power management features the device supports. This includes which power states the device supports, whether or not the device supports waking the machine and from what power states it is capable of wakeup.

3.8.6 Set Power State

The Set Power State operation is used by the operating system to put a device into one of the four power states. The operating system will track the state of all devices in the system.

3.8.7 Get Power Status

The Get Power Status operation is used by the operating system to determine the present state of the power configuration (power states & features). This operation will read the **PMC** and **PMCSR** to obtain this information. Software assumes that reported status information reflects the current state of the function. Therefore functions should update status information bits ONLY after carrying out the intended task.

3.8.8 System BIOS Initialization

Since the PCI Power Management Specification was written to support Operating System directed power management, there are minimal changes required by the system BIOS of the host system. While the system BIOS does not take an active role in power managing the PCI devices, the BIOS is still responsible for basic system initialization. PCI Power Management aware systems should have the following support in the system BIOS Power On Self Test routines:

1. During a true Power-On situation, the BIOS should mask off the **PME#** signal through whatever system specific method is available. This is required because there is a small but finite chance that

a **PME#** event might occur during the power up processes, before a **PME#** handler has been loaded.

2. During a Warm-boot (Control-Alt-Del) sequence, the BIOS should likewise mask off **PME#** events until such as time as the **PME#** event handler loads. This is required because there is a small but finite chance that a device enabled for **PME#** wakeup might have cause to generate a **PME#** during the power up processes, before a **PME#** handler has been loaded.
3. During a Warm-boot (Control-Alt-Del) sequence, the BIOS must restore all PCI Power Managed functions to ***D0***. Functions placed in the ***D1***-***D3*** states will remain in that state until programmed otherwise, or until a PCI reset (**RST#**) occurs. PCI **RST#** assertion is the preferred method for causing all PCI functions to transition to D0. If this cannot be accomplished then the BIOS must individually program each of the PCI functions' **PMCSR** *PowerState* fields to ***D0***.

3.9 Other Considerations

In addition to supporting the minimum set of required mechanisms defined in this specification, designers of PCI and CardBus devices and systems are encouraged to add additional power management functionality, taking advantage of the optional headroom provided by this specification.

Designers are encouraged to design for low power consumption in all operating modes. The PCI Mobile Design Guide makes several suggestions on designing for low power which are applicable to all devices. Even simple things such as minimizing current drain through pull-up resistors can add up to real power savings.

Additional power saving techniques while in ***D0*** are also encouraged as long as they are transparent to the operating system.

©2001 PCMCIA/JEITA

4. PCI-TO-CARDBUS BRIDGE REGISTER DESCRIPTION

4.1 Scope

The document describes the generic functionality and registers needed to implement a CardBus bridge. It is not intended to be implementation specific, nor does it provide much information on the 16-bit portion of the socket interface. This document only touches on the 16-bit PC Card interface sufficiently to describe the configuration spaces required to access the 16-bit PC Card control and status registers.

The primary goal of this chapter is to aid in development of CardBus bridges that have some level of software interface commonality. It is also not intended to detract from, or restrict a designer's efforts to add value. This document assumes an audience versed in the ***PC Card Standard*** and ***PCI Local Bus Specification***.

4.2 Overview

The device described here is a bridge between a PCI bus and two CardBus/16-bit PC Card sockets. Each socket will accept a 16-bit PC Card or CardBus PC Card. At insertion, the bridge detects the type of card installed and provides the functions necessary to support that card. These functions include the needed bus hardware protocol (16-bit PC Card vs. CardBus), the bridge configuration logic needed and power control for the socket.

4.2.1 Assumptions

- Data transfers, from any bus - to any bus, are supported. CardBus cards have access, via the bridge, to other CardBus or 16-bit PC Cards, as well as to the PCI bus.
- Hardware support for cacheable memory residing on a secondary bus device is not provided. If a hardware mechanism was implemented, allowing direct card to card transfers would require propagating addresses to the primary bus to allow snooping. This has a significant impact on CardBus latency and bandwidth, and on unrelated transfers on the primary PCI bus. Software mechanisms should, instead, be used to allow caching of data stored on CardBus cards.
- The bridge provides support for VGA devices. This support is defined in the ***PCI to PCI Bridge Specification***.

4.3 Functional Description

The bridge appears on the primary PCI bus as a multifunction 16-bit PC Card/CardBus bridge with one or two shared sockets. Function 0 provides the 16-bit PC Card and CardBus interface functionality for socket 0. Function 1 contains the same functionality for socket 1.This dividing of configuration and control for each CardBus socket follows the conventions defined for PCI to PCI bridges by the ***PCI to PCI Bridge Architectural Specification***. Using this method allows the bridge to be treated as two independent and separate bridges.

©2001 PCMCIA/JEITA

A bridge designer cannot know how many of the sockets that are available on his part will be used in all system implementations. A bridge ASIC which can provide two sockets may be installed in a system with only one socket used. A method must be provided to tell the bridge how many sockets have been implemented. If a method is not provided, and status retained, software will incorrectly assume and report resources that do not, in fact, exist. Software cannot tell the difference between an empty socket and a non-existent one. Only those configuration spaces relevant to a specific installation should be visible to software. This means, that when configured as a single socket bridge, only the configuration space, and control registers, for the implemented socket should be visible to software. The unimplemented socket's configuration space should be invisible and inaccessible by software. This allows configuration software to deal with a given system implementation without external information as to implemented, versus unimplemented, sockets.

Bus protocol is hardware's responsibility. The bridge must use the correct protocol, on its secondary side, based on what kind of card is plugged in. When a 16-bit PC Card or CardBus card is installed in a socket, the bridge automatically uses that protocol to communicate with the bus. Software is not responsible for setting this up. The functionality to allow software to change protocols is not present.

Memory address mappings for the CardBus bridge are on 4 Kbyte boundaries with a minimum mapping of 4 Kbytes. Cards may be mapped anywhere within the address space assigned to the bridge. The bridge provides two memory base and limit register pairs which may be used for mapping memory mapped I/O or prefetchable memory space.

Two I/O mapping register pairs are provided for each socket. This allows some fragmenting of I/O space on a card and interleaving of I/O space with other I/O devices.

The bridge determines the type of card and Vcc voltages it requires via the Card Detect and VS pins. The Card Detect pins, card type and required/allowed voltages are recorded in the bridge Socket Status Register.

The bridge controls the power to each socket. It keeps any socket not having a card installed powered down. It must automatically remove power from a socket when the resident card is removed. Power must be applied by software. The bridge must not allow a card to be powered at a Vcc voltage that the card does not indicate it can handle via its VS decode.

NOTE The ***PC Card Standard*** explicitly states that a CardBus bridge must not apply a Vcc voltage that is not indicated by the VS decode on a CardBus card (see the ***Electrical Specification***, Card Insertion and Removal). The Standard does not have the same restriction for 16-bit PC Cards. The Standard states that software may change the Vcc voltage based on information found in the CIS without restricting it, as it does for CardBus.

4.4 Bridge Functions

4.4.1 Data Paths

4.4.1.1 Buffer Control

The ***PCI Local Bus Specification*** requires that a device's master and slaves be independent. This means the bridge cannot refuse to accept data from a target bus while trying to move data to it. This is to prevent deadlocks. The CardBus controller must not attempt to rely on the other PCI device backing off. The other device might be a second, identical, CardBus controller making the same incorrect assumption.

 ©2001 PCMCIA/JEITA

4.4.1.2 Parity

The bridge may pass data parity bits that it receives rather than generating them. This is not true when the bridge's internal configuration or I/O spaces are being read by the primary PCI bus. The bridge must then generate appropriate parity. The bridge must also generate parity on all buses for addresses. This is necessary because the address may not be the same value as it initially received. This happens whenever the bridge has received a disconnect indication from a target while still possessing data to transfer. This requires the bridge to generate parity for the new address when it restarts the transfer.

Handling of parity errors by the bridge on its secondary buses is controlled by the Parity Error Response Enable bit in the Bridge Control Register.

4.4.1.3 Locks

Locked accesses should be passed through the bridge as a locked access on the targeted bus. The bridge may treat locks as a resource lock. This simplifies bridge design. If a master attempts a locked access and **LOCK#** is busy (not sourced by the bridge) on the target bus, the bridge will indicate Retry on the source bus. The bridge must keep track of which bus is the source of a locked cycle in order to force Retry on a master from another bus attempting a locked access.

All unlocked accesses to a locked resource will result in a Retry indication.

The ***PCI Local Bus Specification*** does not require support of **LOCK#** in upstream transfers. This means any CardBus located downstream from the primary PCI bus cannot expect locked transactions generated by cards to operate properly. Therefore, PCI-to-CardBus controllers do not need to provide support for that functionality.

4.4.1.4 Retry

When the bridge as master on any bus receives a Retry indication from a slave it must remove its **REQ#** from the target bus. It must wait two cycles before reasserting **REQ#**.

If the length of any data phase becomes greater than 16 PCICLKs the bridge must indicate Retry to the initiator. The bridge should deassert **REQ#** on the target bus at the same time as disconnecting on the initiator bus.

If the transaction being terminated is a read, the bridge must deassert **FRAME#** on the target bus as soon as seeing a valid **TRDY#**. It must remove **IRDY#** as soon as seeing another **TRDY#**. The bridge should discard all data accepted after disconnecting from the source. If the terminated transfer is a write the bridge will continue transferring data until its buffers are empty.

The ***PCI Local Bus Specification*** requires that any device that has a transfer terminated with Retry MUST repeat the exact same transfer, even if the data is no longer required by the Initiator. This might occur when prefetching data. Retrying enables PCI Delayed Transactions to complete properly with a minimum of system delay. If a transaction is not retried properly ***PCI Local Bus Specification*** compliant devices will throw away the transaction after an extended delay. Relying on this delay to end prefetching transactions can significantly degrade system performance.

4.4.2 Memory Address Mapping

The bridge provides multiple base and limit range registers for mapping its buses into the system memory map. These registers are used by the bridge to determine if an access on any of its buses is targeted through the bridge. These windows are configured by software according to available memory space and the requirements of a specific card. The bridge operates on a flat memory map

when a CardBus card is installed. The bridge assumes that any address that is not within the ranges of its secondary buses, resides on the primary bus. Software sets up each secondary bus' address ranges.

There are both Prefetchable Memory and Memory Mapped I/O ranges available. Each is a contiguous address space. The separate ranges need not be contiguous to each other. The bridge uses the values in the mapping registers to determine where the targeted address must lie. If it is within a secondary bus' range it will be claimed by the bridge and passed to the appropriate bus. Accesses by a card to an address that does not reside within its own bus's windows will be passed to the PCI bus, unless the address targeted resides on the other CardBus. In that case it will pass the access to the other secondary bus. CardBus controllers providing more than a single card slot, must support card-to-card transfers.

4.4.3 Bridge Arbitration

The bridge provides arbitration for both secondary buses. This arbitration is based on the principle that all accesses on the primary PCI bus have a higher priority than the secondary buses. Arbitration between the secondary buses is fair. Arbitration is accomplished by looking at **REQ#** on the secondary buses and **FRAME#** on the primary PCI bus. When the bridge is unavailable to the primary bus it must use Retry to indicate it. The bridge uses the **CGNT#** signals to control access on its secondary buses.

Arbitration for the bridge is such that when a collision occurs the primary bus wins. If **FRAME#** is asserted on both the secondary bus and primary PCI on the same cycle and if the address decode of the primary PCI access is within a bridge window, the bridge must retry the secondary bus and reassign **CGNT#** to itself.

Care must be taken in the design of the bridge to avoid the possibility of deadlocks. The PCI specification requires that master and slave interfaces MUST be independent. This means that a master who wishes to move data is not allowed to refuse to accept an access, by another device, before gaining access to the bus itself. If it does it will end up being involved in deadlocks.

Below are some scenarios describing accesses and descriptions of controller actions and responses to avoid deadlocks and move data efficiently.

4.4.3.1 Scenarios

The following scenarios are written for a primary PCI to secondary CardBus. The discussion also applies to transfers from secondary bus to primary or card to card. The singular difference is in the detection of an access request. The bridge uses **FRAME#** on the primary bus and **REQN#** on the secondary buses to detect access attempts. All other issues relate regardless of direction.

4.4.3.1.1 Write

Bridge detects **FRAME#** asserted on the primary bus with an address residing within one of its secondary bus windows.

A. Target Bus Idle: The bridge claims the cycle on the primary bus by asserting **DEVSEL#**. CardBus requires the socket arbitration to be parked at the bridge when the target bus is idle. If the command is a write access the bridge begins to accept data and starts an access on the targeted bus. When the cycle is accepted the bridge will begin to move data. The transfer is ended, normally, when the source master disconnects and the bridge finishes delivering the data to the secondary target. The bridge then disconnects from the target bus. If at any time the bridges buffers fill, and remain so for 16

 ©2001 PCMCIA/JEITA

cycles, the bridge will disconnect from the PCI bus. It must then finish moving the data in its buffers to the target before disconnecting.

B. Target Bus Busy: The bridge claims the access on the primary bus by asserting **DEVSEL#**. The bridge will then indicate Retry to the PCI bus. In the case of a CardBus device requesting an access, the bridge will withhold **CGNT#** until it is no longer busy.

The bridge has two methods of dealing with a write cycle that is not claimed on the target bus. The mode used is controlled by the Master Abort Mode bit in the Bridge Control Register. These methods are described in the ***PCI to PCI Bridge Specification***. Basically, the default mode causes the bridge to continue to accept data as if the cycle had been claimed. It will continue the transfer while discarding the data. This maintains compatibility with previous architectures.

The second mode is to:

1. Signal a target abort to the initiator of the transaction.
2. Set the Received Master Abort status bit for the target bus.
3. Assert **SERR#** on the primary bus, if the aborted transaction was a posted write, and **SERR#** is enabled.

4.4.3.1.2 Read

Reads through the bridge are done using delayed reads. This is necessary to be a ***PCI Revision 2.1*** compliant bridge. Designers should refer to that specification for a detailed description of delayed reads. Simply put reads are treated somewhat like a split transaction. This is to improve system throughput. The simplest way for a bridge to handle reads would be to just hold the requester in wait states. This can cause severe degradation of system bandwidth for machines having multiple active masters.

To avoid this the bridge will save the address and command, capture the byte enables and then indicate Retry to the primary bus. When the target bus is available it will begin a read on the target bus. The bridge will continue to indicate Retry to the Source master until it has obtained the data requested. All other read accesses to the bridge on the primary bus will also be Retried. When the bridge has the requested data, it will accept a Read that matches the original address and present the buffered data. If the data is in a prefetchable window it will continue to read data until either its buffers are full or the source master completes the read with a normal disconnect. All data remaining in its buffers, unread, will be discarded.

When a read transaction results in the bridge signaling a Master Abort, the default operation is to return FFFF FFFFH to the initiator with **TRDY#**. The transaction is ended normally. When the Master Abort Mode bit is set the bridge ends the transaction the same way it does in the case of a write. See ***4.4.3.1.1 Write*** above.

In any case, the bridge will not allow a burst, Read or Write, to continue beyond the end of its prefetchable limit register.

4.4.4 Interrupts

CardBus interrupts must be routed through the bridge to PCI INT lines. Two lines, **INTA** and **INTB**, should be provided for PCI interrupts. While a single PCI interrupt may be used, this may reduce performance. IREQ interrupts should, by default, be routed to the PCI INT lines. This will allow a 16-bit interface application to use shareable interrupts by upgrading their driver. This adds to the ease of use and compatibility of a 16-bit PC Card across differing system configurations. If a controller

provides IRQ routing, Bit #7 of the Bridge Control Register should be used to select between the IRQ and INT routing registers.

CSTSCHG interrupts should be routed to a PCI INT line. Card and Socket Services is the recipient of this interrupt. Allowing routing to IRQ pins is not required for this signal.

Routing of PCI style interrupts is accomplished via the Interrupt Line Register in configuration space. IRQ routing is controlled by the 16 bit control registers when an 16-bit PC Card is installed and IRQ routing is enabled.

CardBus interrupts are required to use INT signals on the PCI bus, 16-bit PC Card interrupts use legacy IRQ signals. CardBus bus host adapters must provide both to ensure compatibility with existing and future applications and system software. System vendors using these controllers should take advantage of the interrupts provided to avoid compatibility problems with their systems. Relying on non-standard software modifications to compensate for missing interrupt lines will result in non-standard implementations and may result in unpredictable reliability problems with some system / card / application combinations.

4.4.5 Reset

The reset signal for CardBus (**CRST#**) and 16-bit (**RESET**) interfaces share the same pin, but have inverted polarities. This requires the bridge to control the polarity of this pin based on the card type installed.

All registers in the bridge are initialized to their default values by **PCIRST#**.

The bridge must drive **CRST#** when:

1. **PCIRST#** is asserted. **CRST#** will be released when **PCIRST#** is.
2. Whenever a socket is powered down and the bridge is powered up.
3. When bit #6 of the Bridge Control Register is written by software. **CRST#** will be released when software specifically clears the control bit *and* 256 PCICLKs have passed, since it was asserted.

The bridge handles attempted accesses to the card, while **CRST#** is asserted, in two different manners.

1. In the case where **CRST#** is asserted and software has cleared the **CRST#** bit in the Bridge Control Register, but the 256 cycle counter has not run out, the bridge will Retry all accesses until **CRST#** is released.
2. In the case where **CRST#** is asserted and software has NOT cleared the **CRST#** bit, the bridge will respond to any access targeted at the card in the manner proscribed by the Master Abort Mode bit in the Bridge Control Register.

Asserting **CRST#** in this manner will cause the bridge to clear the *CSTSCHG* bit in the Socket Status Register associated with the CardBus socket. All other bits in the Socket Status Register will remain unchanged unless the status causing them is changed. The *CSTSCHG* bit will be again set if it is driven by the card after **CRST#** is deasserted.

4.4.6 ISA Mode

ISA mode prevents the bridge from forwarding to a CardBus any I / O transaction that addresses the top 768 bytes of each naturally aligned 1 Kbyte block within the defined I / O range. It also causes the bridge to pass to the primary bus any CardBus I / O access to the top 768 bytes in that same range.

©2001 PCMCIA/JEITA

This limits devices, on the secondary bus, to using the first 256 bytes of the I/O range when this mode is enabled.

ISA Mode Addressing Map For Any 4K Byte Block	
I/O Addresses Residing on Primary (PCI) Bus	**I/O Addresses Residing on Secondary (CardBus) Bus.**
0 nD00 - 0 nFFFH	0 nC00 - 0 nCFFH
0 n900 - 0 nBFFH	0 n800 - 0 n8FFH
0 n500 - 0 n7FFH	0 n400 - 0 n4FFH
0 n100 - 0 n3FFH	0 n000 - 0 n0FFH

4.4.7 Support for VGA devices at ISA addresses

The bridge provides support for ISA compatible addressing of downstream CardBus VGA devices. When the *VGA Enable* bit is set in the Bridge Control Register the bridge will positively decode and forward accesses on its primary bus in the range 000A 0000H through 000B FFFFH. In this mode it will not forward accesses to this range on the enabled secondary bus. The bridge will also forward I/O addresses in the ranges 3B0H through 3BBH and 3C0 through 3DFH. All aliases of these addresses (AD[15::10] are not decoded) are also passed. Refer to the ***PCI to PCI Bridge Specification***, ***Appendix A*** for a complete description of this and the VGA palette snooping functionality.

VGA palette snooping is supported via bit #5 of the Bridge Command Register. This causes the bridge to accept I/O writes to addresses 3C6H, 3C8H, and 3C9H. Reads from these addresses are ignored by the bridge unless the *VGA Enable* bit is set in the Bridge Control Register.

4.5 Configuration

Logically the bridge looks to the primary PCI as two separate secondary buses residing in a single device. Each socket has its own configuration space. This makes the bridge a multifunction device.

Bridge configuration space is accessible only from the primary PCI bus. No other interfaces respond to configuration cycles.

Configuration commands on the primary PCI bus will be handled in the following manner:

Type 0 (**AD[1::0]** = 00): If selected by IDSEL the bridge will decode bits 07:02 to determine the specific Dword configuration register being selected. Based on the configuration command (Read/Write) and the **C/BE[3::0]#** lines the bridge will provide data from selected register or write the data proffered. Read data will be all 32 bits of the Dword register, regardless of byte enables, with the requested data driven in its natural byte location. Write data will be deposited into the selected register using the **C/BE[3::0]#** lines to enable the write.

Type 1 (**AD[1::0]** = 01): If the Bus Number (bits 23:16) matches the controller's secondary bus number it will initiate a Type 0 configuration cycle on the targeted bus. It sets **AD[1::0]** = 00 and passes through **AD[10::2]** unchanged. The configuration cycle data will be presented (or accepted in the case of a Read) during the data phase of the transaction.

Type 1 cycles that do not match the bridges secondary bus but do match a subordinate bus will be passed unmodified.

Type 0 configuration cycles on the controller's subordinate bus are ignored by the controller. Type 1 commands on the secondary bus are also ignored except in the case where the Device Number is all 1's. Refer to the discussion of Special Cycles for handling of this case.

4.5.1 Bridge Configuration Registers

The bridge has two separate configuration spaces. There is one implemented for each socket.

When configured as a two slot CardBus bridge, the bridge has two configuration spaces. Function 0 is for socket #0. Function 1 is the CardBus configuration for slot #1. CardBus has been defined as closely as possible to PCI in order to enable common silicon. The definition of the configuration spaces for CardBus has been kept close to that defined for PCI to PCI bridges. Differences in I/O Register Base and Limit Registers are driven by CardBus needing 4 byte resolution whereas PCI to PCI uses 4 Kbyte resolution.

The following tables list the bridge configuration registers. Some registers are not unique to a function or slot. These include Vendor ID, Header Type, etc. Some bridge registers are only used by some functions or one socket.

Table 4-1: CardBus Controller Configuration Space

Device ID = nnnn		Vendor ID = nnnn		00H
Status		Command		04H
Class Code = 060700H			Revision ID = nn	08H
BIST	Header Type = 82H	Latency Timer	Cache Line Size	0CH
PC Card Socket Status and Control Registers Base Address				10H
Secondary Status		Reserved	Cap_Ptr	14H
CardBus Latency Timer	Subordinate Bus Number	CardBus Bus Number	PCI Bus Number	18H
Memory Base 0				1CH
Memory Limit 0				20H
Memory Base 1				24H
Memory Limit 1				28H
I/O Base 0 (Upper 16 Bits, optional)		I/O Base 0 (Lower 16 Bits)		2CH
I/O Limit 0 (Upper 16 Bits, optional)		I/O Limit 0 (Lower 16 Bits)		30H
I/O Base 1 (Upper 16 Bits, optional)		I/O Base 1 (Lower 16 Bits)		34H
I/O Limit 1 (Upper 16 Bits, optional)		I/O Limit 1 (Lower 16 Bits)		38H
Bridge Control		Interrupt Pin	Interrupt Line	3CH
Subsystem ID (optional)		Subsystem Vendor ID (optional)		40H
PC Card 16 Bit IF Legacy Mode Base Address (Optional)				44H
Reserved				48-7FH
User Defined				80-FFH

©2001 PCMCIA/JEITA

4.5.2 PCI-PC Card Bridge Configuration Registers

All of the registers in the predefined header space comply with the functionality described in the ***PCI Local Bus Specification***.

4.5.2.1 Vendor ID (Offset = 00H)

Default	Description
nnnnh	The Vendor ID identifies the manufacturer of the device. Vendor identifiers are allocated by the PCI SIG.

4.5.2.2 Device ID (Offset = 02H)

Default	Description
nnnnh	The Device ID identifies the particular device. The identifier is allocated by the vendor.

4.5.2.3 Command Register (Offset = 04H)

Default	Description
0000H	The bits in the Command Register adhere to the definitions set out in the ***PCI Local Bus Specification*** and the ***PCI to PCI Bridge Specification***.

4.5.2.4 Status Register (Offset = 06H)

Default	Description
nnnnh	The Status Register is in the portion of the predefined header that is common to all PCI devices. The bits in this register adhere to the definitions in the ***PCI Local Bus Specification***, but only apply to the primary PCI interface.

4.5.2.5 Revision ID (Offset = 08H)

Default	Description
nnh	The Revision ID is selected by the vendor.

4.5.2.6 Class Code (Offset = 09H)

Default	Description
060700H	The Class Code is 06 (bridge device) 07 (CardBus) 00 (programming interface)

4.5.2.7 Cache Line Size (Offset = 0CH)

Default	Description
00H	This register conforms to the definition specified in the ***PCI Local Bus Specification***.

4.5.2.8 Latency Timer (Offset = 0DH)

Default	Description
00H	This register adheres to the ***PCI Local Bus Specification*** but applies only to the primary interface.

4.5.2.9 Header Type (Offset = 0EH)

Default	Description
82h	PCI Header Type for a multi-socket and/or multifunction PCI-to-CardBus bridge.
02H	PCI Header Type for a single socket PCI-to-CardBus bridge.

4.5.2.10 BIST (Offset = 0FH)

Default	Description
00H	The bits in this register adhere to the definitions in the ***PCI Local Bus Specification***. If a BIST is not implemented this register should be read only and always return zeros.

4.5.2.11 PC Card Socket Status and Control Registers Base Address (Offset = 10H)

Default	Description
0000 0000H	This register points to the memory mapped I/O space that contains both the CardBus and 16-bit Status and Control registers. CardBus Status and Control registers start at offset 000H and the 16-bit card registers begin at offset 800H. Bits [31::11] are R/W. Bits [11::00] are hardwired to zero. This indicates to configuring software that the bridge wants 4K bytes of non-prefetchable memory space, starting on a 4K boundary, that can be mapped anywhere in memory. This register's bits adhere to the definitions set out in the ***PCI Local Bus Specification***.

4.5.2.12 Cap_Ptr (Capabilities Pointer) (Offset = 14H)

Default	Description
Implementation Specific	Provides an offset into the function's PCI Configuration Space for the location of the first item in the Capabilities Linked List. The Cap_Ptr offset is DWORD aligned so the two least significant bits are always "0"s. The Cap_Ptr has a minimum value of 80H and a maximum value of 0F8H if implemented. If the New Capabilities feature is not implemented, this byte reads "00".

©2001 PCMCIA/JEITA

4.5.2.13 Reserved (Offset = 15H)

Default	Description
00H	

4.5.2.14 Secondary Status (Offset = 16H)

Default	Description
0000H	The Secondary Status Register is similar in function to the Primary Status Register but contains information relating to the CardBus. Bit 14 of this register is defined differently than the Primary. When set it indicates that the bridge has detected **SERR#** asserted on the CardBus. This function is identical to that specified in the ***PCI to PCI Bridge Specification***.

4.5.2.15 PCI Bus Number (Offset = 18H)

Default	Description
00H	The Primary Bus Number identifies the number of the PCI bus on the primary side of the bridge. This is set by the appropriate configuration software.

4.5.2.16 CardBus Bus Number (Offset = 19H)

Default	Description
00H	The CardBus Number identifies the number of the CardBus attached to the socket. This is set by PCI BIOS configuration software or Socket Services software. This register is called the “Secondary Bus Number” on a PCI to PCI bridge.

4.5.2.17 Subordinate Bus Number (Offset = 1AH)

Default	Description
00H	The Subordinate Bus Number is a register defined for PCI to PCI bridges. It holds the number of the bus at the lowest part of the hierarchy behind the bridge. Normally, a CardBus bridge will be at the bottom of the bus hierarchy and this register will hold the same value as the CardBus Bus Number register.

4.5.2.18 CardBus Latency Timer (Offset = 1BH)

Default	Description
00H	The CardBus Latency Timer has the same functionality of the primary PCI bus Latency Timer but applies to the CardBus attached to this specific socket. This is set by PCI BIOS configuration software or Socket Services software.

4.5.2.19 Memory Base #0 (Offset = 1CH)

Default	Description
0000 0000H	The Memory Base Register defines the bottom address of a memory mapped I/O window. The upper 20 bits correspond to address bits **AD[31::12]**. The bottom 12 bits of this register are read only and return zero when read. This window is enabled by bit #1 of the Command Register. Prefetching within this window is controlled by bit #8 of the Bridge Control Register. The default is enabled and should only be cleared if Memory Reads will cause side effects on the installed card.

4.5.2.20 Memory Limit #0 (Offset = 20H)

Default	Description
0000 0000H	The Memory Limit Register defines the top address of the memory mapped I/O space. The upper 20 bits of this register correspond to **AD[31::12]**. The bottom 12 bits of this register are read only and return zeros when read. The bridge assumes the bottom address bits [11:0] are ones to determine the range defined. So if the Memory Base and Limit registers are set to the same value a window of 4 KBytes is defined. Both Memory windows are both enabled by Command Register Bit #1. To disable either window individually, the Limit register of that range should be set below the Base. This will cause the bridge to never detect a hit on that window.

4.5.2.21 Memory Base #1 (Offset = 24H)

Default	Description
0000 0000H	Memory Base 1 has the same functionality as Memory Base 0. This window is enabled by bit #1 of the Command Register. Prefetching within this window is controlled by bit #9 of the Bridge Control Register. The default is enabled and should only be cleared if Memory Reads will cause side effects on the installed card.

4.5.2.22 Memory Limit #1 (Offset = 28H)

Default	Description
0000 0000H	The Memory Limit 1 register has the same functionality as the Memory Limit 0 register.

©2001 PCMCIA/JEITA

4.5.2.23 I/O Base #0 (Lower 16 Bits) (Offset = 2CH)

Default	Description
0000H	The IO Base Register defines the bottom address of an address range that is used by the bridge to determine when to forward an I/O transaction to the CardBus. The bits in this register correspond to **AD[15::0]**. Bits **AD[1::0]** are used to indicate whether the bridge implements 16 or 32 bit I/O addressing. For address decoding, if only 16 bit addressing is implemented the bridge must determine that the upper 16 bits of address are zeros before accepting an access. **I/O Base[1::0]** / **I/O Addressing Capability** 00H – 16 bit I/O addressing 01H – 32 bit I/O addressing Address bits **AD[15::2]** provide the 4 byte granularity required by CardBus. This I/O mapping varies from a PCI to PCI bridge, in that it allows mapping the windows on a 4 byte boundary with a minimum size of 4 bytes. A PCI to PCI bridge maps I/O windows on 4K boundaries with a minimum 4 Kbyte size.

4.5.2.24 I/O Base #0 (Upper 16 Bits) (Offset = 2EH)

Default	Description
0000H	This optional register is an extension to the I/O Base Register. It defines bits **AD[31::16]**. If this and the I/O Limit Upper 16 Bits Register are not implemented, I/O devices behind the bridge will all be mapped below 0001 0000H and the bridge must validate that bits **AD[31::16]** are all set to zero before accepting an access.

4.5.2.25 I/O Limit #0 (Lower 16 Bits) (Offset = 30H)

Default	Description
0000H	The I/O Limit Register defines the top address of the address range that is used by the bridge to determine when to forward an I/O access to the CardBus. The bits in this register correspond to **AD[15::00]**.

4.5.2.26 I/O Limit #0 Upper 16 Bits (Offset = 32H)

Default	Description
0000H	This optional register is an extension to the I/O Limit Register.

4.5.2.27 I/O Base #1 (Lower 16 Bits) (Offset = 34H)

Default	Description
0000H	The IO Base Register defines the bottom address of an address range that is used by the bridge to determine when to forward an I/O transaction to the CardBus. The bits in this register correspond to A**D[15::0]**. Bits **AD[1::0]** are used to indicate whether the bridge implements 16 or 32 bit I/O addressing. For address decoding, if only 16 bit addressing is implemented the bridge must determine that the upper 16 bits of address are zeros before accepting an access. Address bits **AD[15::2]** provide the 4 byte granularity required by CardBus. This I/O mapping varies from a PCI to PCI bridge, in that it allows mapping the windows on a 4 byte boundary with a minimum size of 4 bytes. A PCI to PCI bridge maps I/O windows on 4K boundaries with a minimum 4 Kbyte size.

4.5.2.28 I/O Base #1 (Upper 16 Bits) (Offset = 36H)

Default	Description
0000H	This optional register is an extension to the I/O Base Register. It defines bits **AD[31::16]**. If this and the I/O Limit Upper 16 Bits Register are not implemented, I/O devices behind the bridge will all be mapped below 0001 0000H and the bridge must validate that bits **AD[31::16]** are all set to zero before accepting an access.

4.5.2.29 I/O Limit #1 (Lower 16 Bits) (Offset = 38H)

Default	Description
0000H	The I/O Limit Register defines the top address of the address range that is used by the bridge to determine when to forward an I/O access to the CardBus. The bits in this register correspond to **AD[15::00]**.

4.5.2.30 I/O Limit #1 (Upper 16 Bits) (Offset = 3AH)

Default	Description
0000H	This optional register is an extension to the I/O Limit Register.

4.5.2.31 Interrupt Line (Offset = 3CH)

Default	Description
nnh	This register is used in the manner defined in the ***PCI Local Bus Specification*** and ***PCI to PCI Bridge Specification***.

4.5.2.32 Interrupt Pin (Offset = 3DH)

Default	Description
Default Socket 0: 01H Default Socket 1: 10H or 01H	Read only register. Bit definition adheres to the ***PCI Local Bus Specification*** and ***PCI to PCI Bridge Specification***.

©2001 PCMCIA/JEITA

4.5.2.33 Bridge Control Register (Offset = 3EH)

The Bridge Control register provides extensions of the Command Register that are specific to PCI to PCI and PCI-to-CardBus bridges.

Default Value	Bit #	Description
0000H	0	*Parity Error Response Enable* - Controls the response to parity errors on the CardBus. When 0 parity errors are ignored. When set parity on the CardBus is checked and errors reported. Default on reset is zero (disabled).
	1	*SERR# Enable* - Controls forwarding of **SERR#** signals indicated on the CardBus. When set the bridge will forward to the PCI bus a **SERR#** indication on the CardBus. Default on reset is zero (disabled).
	2	*ISA Enable* - This applies only to addresses that are enabled by the I/O Base and Limit registers and are also in the first 64 KBytes of PCI I/O space. When set the bridge will block forwarding from PCI-to-CardBus I/O transactions addressing the last 768 byte in each 1 KByte block. In the opposite direction (CardBus to PCI) I/O transactions will be forwarded if they address the last 768 byte in each 1 K block. Default on reset is zero (disabled).
	3	*VGA Enable* - Modifies the bridge's response to VGA compatible addresses. When the *VGA Enable* bit is set the bridge will forward transactions in the following ranges: Memory: 0A 0000H to 0B FFFFH I/O: Addresses where AD[9:0] are in the ranges 3B0H to 3BBH and 3C0H to 3dFH (inclusive of ISA address aliases - AD[15:10] are not decoded). When the *VGA Enable* bit is set forwarding of these accesses is independent of both the I/O and memory address ranges. Forwarding is also independent of the setting of the *ISA Enable* bit in the **Control** register or the *VGA Snoop* bit in the **Command** register. Forwarding of these accesses IS affected by the *I/O* and *Memory Enable* bits in the **Command** Register. Default on reset is zero (disabled).
	4	Reserved - Returns zero on read.
	5	*Master Abort Mode* - Controls the behavior of the bridge when a master abort occurs on either PCI or CardBus interface when the bridge is master. When not set the bridge must return all ones if a master abort occurs during a read. During a write the data should be dropped in the bit bucket. When set the bridge signals a target abort to the requesting master when the corresponding transaction on the opposite bus terminates with a master abort and the transaction on the source side is not complete (reads and non-posted writes). If the transaction on the source bus is complete, and **SERR#** is enabled in the **Command** register, the bridge must assert **SERR#** on the PCI bus. Default on reset is zero.
	6	*CardBus Reset* - When set the bridge will assert and hold **CRST#**. When cleared the bridge will deassert **CRST#**. This bit may be set by software. It will also be set by hardware when the controller executes the powerdown sequence. This bit is cleared only by software. **CRST#** is a wire-OR of this control bit and **PCIRST#**.
	7	*IREQ-INT Enable* - When set this bit enables the IRQ routing register for 16-bit PC Cards. When cleared IREQ interrupts will be routed to the INT pin indicated by the Interrupt Pin Register. This bit should be cleared by **PCIRST#**
	8	*Memory 0 Prefetch Enable* - When set enables Read prefetching from the memory window defined to by the Memory Base 0 and Memory Limit 0 registers. Default on reset is one (enabled).
	9	*Memory 1 Prefetch Enable* - When set enables Read prefetching from the memory window defined to by the Memory Base 1 and Memory Limit 1 registers. Default on reset is one (enabled).
	10	*Write Posting Enable* - Enables posting of Write data to and from the socket. If this bit is not set the bridge must drain any data in its buffers before accepting data for or from the socket. Each data word must then be accepted by the target before the bridge can accept the next from the source master. The bridge must not release the source master, until the last word is accepted by the target. Operating with write posting disabled will inhibit system performance.
	11-15	Reserved - Return zero when read.

4.5.2.34 Subsystem Vendor ID (Offset = 40H)

Default	Description
0000H	This optional register is identical to that described in the ***PCI Local Bus Specification***. Must return zeros if not implemented. This identifier is assigned by the PCI SIG. It is necessary to implement this register for if a CardBus controller is used on a third party product (add-in board or subsystem). It provides a mechanism for system software to uniquely identify the subsystem vendor of the product.

4.5.2.35 Subsystem ID (Offset = 42H)

Default	Description
0000H	This optional register is identical to that described in the ***PCI Local Bus Specification***. Must return zeros if not implemented. It is necessary to implement this register if a CardBus controller may be used on a third party product (add-in board or subsystem). It provides a mechanism for system software to identify the product. This value is vendor specific.

4.5.2.36 PC Card 16 Bit IF Legacy Mode Base Address (Optional) (Offset = 44H)

Default	Description
0000 0001H	This optional register points to the index and data registers that resided at 3E1 and 3E0 in the 82365. This register is intended only for legacy mode operation. It is not recommended that this mode be used by new software. New code should use the PC Card Socket Status and Control Registers Base Address space to address the registers directly. This register is cleared and disabled by **PCIRST#**. It must not respond to PCI cycles unless specifically loaded with a non-zero address after **PCIRST#** is deasserted. When this register is enabled memory mapped accesses to the Socket and Control Registers, via PC Card Socket Status and Control Registers Base Address Register, are disabled. This makes the usage of this mode and the memory mapped mode mutually exclusive. This register's bits adhere to the definitions set out in the ***PCI Local Bus Specification***.

Note: The implementation of this optional register does not remove the requirement to support direct memory mapped access to the 16 bit PC Card interface Control and Status registers via the PC Card Socket Status and Control Registers Base Address at 10H.

4.5.3 Socket Status & Control Registers

These registers must be mapped into memory space to allow faster access than would be possible if they were in configuration space. All of these registers shall be initialized by **PCIRST#**.

Where functions in the CardBus and 16-bit registers overlap, the CardBus registers should take precedence. Redundant bits and / or registers in 16-bit space can be deleted, when the functionality is also present in the CardBus registers, unless you have implemented the Legacy Mode Base Address Register to allow usage of existing point enabler software. These registers must be accessible to software regardless of the card type installed in the socket.

©2001 PCMCIA/JEITA

Table 4-2: Status & Control Registers

CardBus Registers				Offset
Event				00H
Mask				04H
Present State				08H
Force				0CH
Control				10H
Reserved				14-1FH
User Defined				20-7FF
16-Bit Registers				
Interrupt and General Control	Power & RESETDRV Control	I/F Status	ID and Revision	800H
I/O Control	Address Window Enable	Card Status Change Interrupt Configuration	Card Status Change	804H
I/O Add. 0 Stop High Byte	I/O Add. 0 Stop Low Byte	I/O Add. 0 Start High Byte	I/O Add. 0 Start Low Byte	808H
I/O Add. 1 Stop High Byte	I/O Add. 1 Stop Low Byte	I/O Add. 1 Start High Byte	I/O Add. 1 Start Low Byte	80CH
Sys. Mem. Add 0 Mapping Stop High Byte	Sys. Mem. Add 0 Mapping Stop Low Byte	Sys. Mem. Add 0 Mapping Start High Byte	Sys. Mem. Add 0 Mapping Start Low Byte	810H
User Defined	User Defined	Card Memory Offset Add. 0 High Byte	Card Memory Offset Add. 0 Low Byte	814H
Sys. Mem. Add 1 Mapping Stop High Byte	Sys. Mem. Add 1 Mapping Stop Low Byte	Sys. Mem. Add 1 Mapping Start High Byte	Sys. Mem. Add 1 Mapping Start Low Byte	818H
User Defined	User Defined	Card Memory Offset Add. 1 High Byte	Card Memory Offset Add. 1 Low Byte	81CH
Sys. Mem. Add 2 Mapping Stop High Byte	Sys. Mem. Add 2 Mapping Stop Low Byte	Sys. Mem. Add 2 Mapping Start High Byte	Sys. Mem. Add 2 Mapping Start Low Byte	820H
User Defined	User Defined	Card Memory Offset Add. 2 High Byte	Card Memory Offset Add. 2 Low Byte	824H
Sys. Mem. Add 3 Mapping Stop High Byte	Sys. Mem. Add 3 Mapping Stop Low Byte	Sys. Mem. Add 3 Mapping Start High Byte	Sys. Mem. Add 3 Mapping Start Low Byte	828H
User Defined	User Defined	Card Memory Offset Add. 3 High Byte	Card Memory Offset Add. 3 Low Byte	82CH
Sys. Mem. Add 4 Mapping Stop High Byte	Sys. Mem. Add 4 Mapping Stop Low Byte	Sys. Mem. Add 4 Mapping Start High Byte	Sys. Mem. Add 4 Mapping Start Low Byte	830H
User Defined	User Defined	Card Memory Offset Add. 4 High Byte	Card Memory Offset Add. 4 Low Byte	834H
User Defined				838H-83FH
Sys. Mem Add 3 Mapping Start Upper Byte (Optional)	Sys. Mem Add 2 Mapping Start Upper Byte (Optional)	Sys. Mem Add 1 Mapping Start Upper Byte (Optional)	Sys. Mem Add 0 Mapping Start Upper Byte (Optional)	840H
User Defined	User Defined	User Defined	Sys. Mem Add 4 Mapping Start Upper Byte (Optional)	844H
User Defined				848-FFFH

4.5.3.1 Event (Offset = 00H)

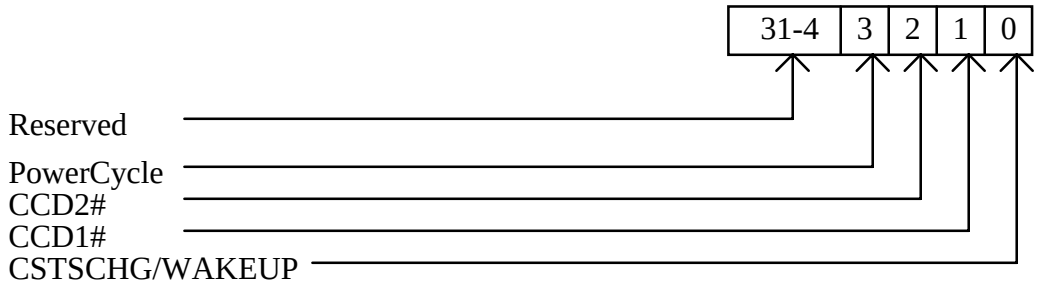


Figure 4-1: Event Register

The Status Event Register indicates a change in socket status has occurred. **CSTCHG#** is driven, by the controller, based on the bits in this register. These bits do not indicate what the change is, only that one has occurred. Software must read the Socket Present State Register for current status. All of the bits in this register can be cleared by writing a one to that bit. These bits can be set to a one by software through writing a one to the corresponding bit in the Force Register. All bits in this register are cleared by **PCIRST#** . They may be immediately set again, if when coming out of **CRST#** the bridge finds the status unchanged (i.e. **CSTSCHG** reasserted or Card Detects still true). Software needs to clear this register before enabling interrupts. If it is not cleared, when interrupts are enabled an interrupt will be generated based on any bit set but not masked.

Default Value	Bit #	Description
00000000H	0	*CSTSCHG/WAKEUP* - Card Status Change and/or Wakeup bit. This bit indicates that the **CSTSCHG** and/or **WAKEUP** signal has been asserted. It only indicates the assertion event. It is not a reflection of the *CSTSCHG* bit from the card. It will be latched by the controller and must be explicitly cleared by the appropriate software. The status change interrupt, driven by the controller, must be based on this event bit rather than the Present Value register. When a card is powered this bit indicates a status change and is driven continuously by the card. When a socket is powered down, this bit is a *WAKEUP* bit. A card might only drive it for 1 ms to limit drain on a battery. To be used in this manner a card must have an external supply or battery. Deassertion of **CSTSCHG** is controlled by software or reset clearing the signal on the bus. Indicating that change would not be useful. This bit will not be set if an event is detected during the time period when the bridge has started the power up cycle of the socket but has not yet signaled a Power Up Complete interrupt. This prevents spurious signals from a card, during powerup, generating invalid events. This bit will be re-enabled when the Power Complete interrupt is generated. During the power down sequence the card is responsible for preventing glitches.
	1-2	*CCD1* and *CCD2* - Card Detects 1 and 2. Indicate a change has occurred in the corresponding Card Detect bit.
	3	*Power Cycle* - The bridge has completed powering up the CardBus socket. The Present State register should be read to determine that the voltage requested was actually applied. The bridge will not allow an unsupported voltage to be applied to a CardBus card. This bit is meaningless when a 16-bit PC Card is installed
	4-31	Reserved

©2001 PCMCIA/JEITA

4.5.3.2 Mask (Offset = 04H)

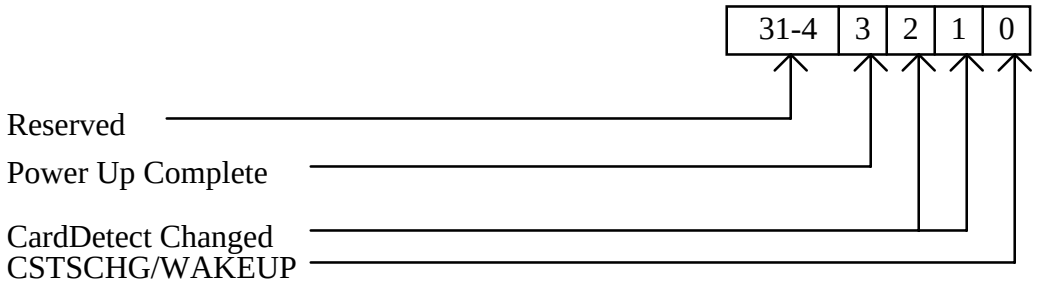


Figure 4-2: Mask Register

This register gives software the ability to control what status change interrupts are generated by the bridge. If the *Card Detect Changed* bit is enabled at the time a card is removed, an interrupt will be generated. Then this register is cleared automatically. This is to prevent spurious interrupts while cards are removed. If it is desired to have the bridge generate an interrupt at the time of a new cards insertion, software must again set the *Card Detect Changed* mask bit. This register is cleared by a **PCIRST#**.

Default Value	Bit #	Description
00H	0	*CSTSCHG/WAKEUP* - When set enables an interrupt based on the **CSTSCHG** signal being asserted by a CardBus card. This bit is meaningless when an 16-bit card is installed. CSTSCHG interrupts generated by 16-bit PC Cards are controlled via registers in that interface's register space. Default = 0 (disabled).
	1-2	*Card Detect Changed* - When set enables an interrupt when the bridge detects change. Default = 00 (disabled).
	3	*Power Cycle Complete* - When set the bridge will generate an interrupt 256 cycles after powering up a socket. Default = 0 (disabled).
	4-31	Reserved

4.5.3.3 Present State (Offset = 08H)

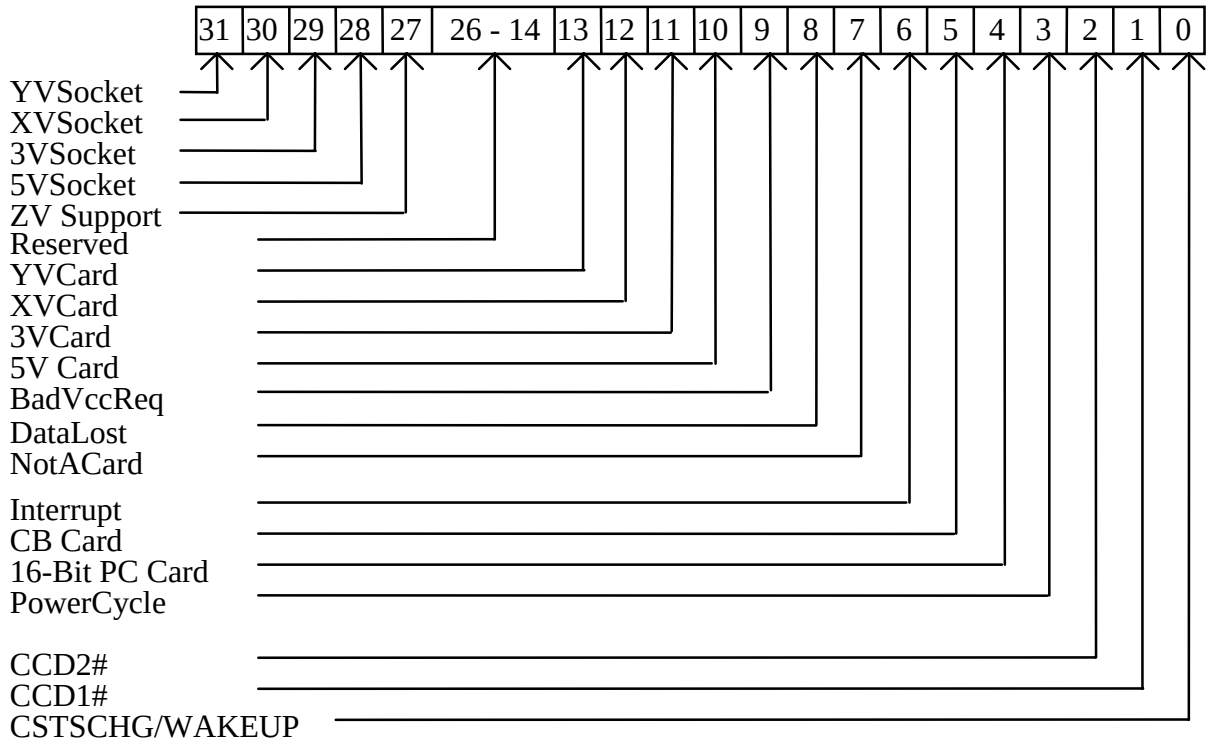


Figure 4-3: Present State Register

The Socket Present State Register reflects the present value of the socket's status. Some of the bits in this register are merely reflections of interface signals while others are flags set to indicate a status change.

©2001 PCMCIA/JEITA

Default Value	Bit #	Description
00000000H	0	*CSTSCHG/WAKEUP* - Card Status Change bit. This bit reflects the current status of the CSTSCHG/WAKEUP pin on the CardBus interface.
	1-2	*CCD1# and CCD2#* - Card Detects 1 and 2. Provide for detection of a PC card insertion/removal/presence. Also used by the bridge, in conjunction with **CVS1** and **CVS2**, to determine card type (16-bit PC Card vs. CardBus). They are reflections of the **CCD1** and **CCD2** pins.
	3	*PowerCycle* - When set indicates the interface is powered up. When cleared the socket is powered down. Set to zero by **PCIRST#**.
	4	*16-bit PC Card* - When set indicates that the Card Detect state machine determined a 16-bit PC Card was inserted. This bit is cleared when another card is inserted that is not 16-bit. This bit is set to zero by a **PCIRST#**.
	5	*CBCard* - When set indicates that the Card Detect state machine determined a CardBus card was inserted. This bit is cleared when another card is inserted that is not CardBus. This bit is set to zero by **PCIRST#**. NOTE: This bit and the *16-bit PC Card* bit do not indicate that a card is installed. They only indicate what kind of card was last installed. The Card Detect bits indicate if a card is currently in the socket.
	6	*Interrupt* - When set to one indicates the inserted card is driving its interrupt pin true. This bit is not a registered bit and its assertion/deassertion must follow the interrupt pin from the card.
	7	*NotACard* - Indicates that an unsupported card is installed in the socket. The bridge will not allow power to be applied to the socket if this bit is set. Set to zero by **PCIRST#**.
	8	*DataLost* - Indicates that a card was removed while the interface was active. Data may have been lost. Any data in the bridge's data buffers when this occurs is lost. Set to zero by **PCIRST#**. WHY: To allow software to fail in a graceful manner, if it chooses to, when this happens.
	9	*BadVccReq* - Software attempted to apply an unsupported or incorrect voltage level to a socket. This bit is set to zero by **PCIRST#**.
	10	*5VCard* - When set the card installed requires and/or supports 5.0V operation. This bit is set by the state machine used to detect card voltage requirements. This bit is set to zero by **PCIRST#**.
	11	*3VCard* - When set the card installed requires and/or supports 3.3V operation. This bit is set by the state machine used to detect card voltage requirements. This bit is set to zero by **PCIRST#**.
	12	*XVCard* - Unused. Returns 0.
	13	*YVCard* - Unused. Returns 0.
	14-26	Reserved - Return zero when read.
	27	ZVSupport – Zoomed Video Support. This bit indicates Whether or not the socket has support for Zoomed Video.
	28-31	*Vcc Voltage Available* - Indicates the Vcc voltages available for the sockets in this machine. **Bit # 31** / **Bit #30** / **Bit #29** / **Bit #28** YV / XV / 3V / 5V

4.5.3.4 Force (Offset = 0CH)

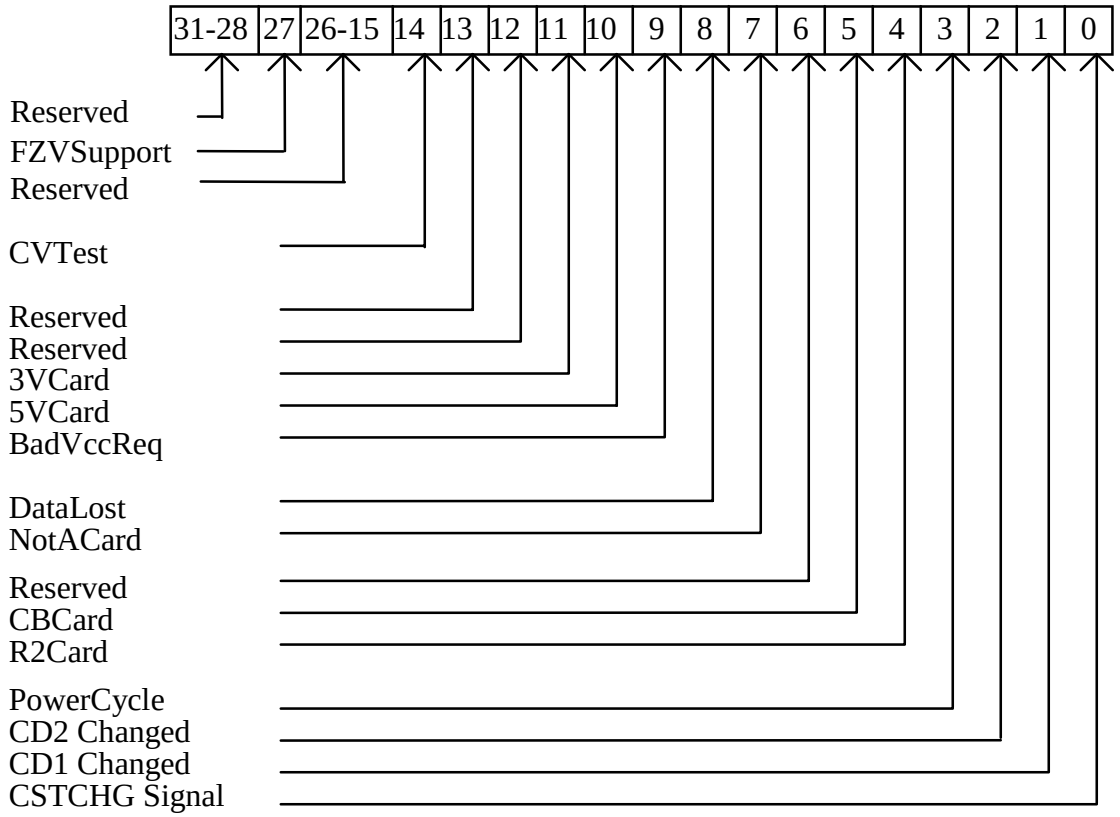


Figure 4-4: Force Register

The Force Register is a phantom register. Its bits are merely control bits. They are not registered and need no clearing. It provides software the ability to force various status and event bits in the bridge. This gives software the ability to test and restore status. Writing a one to a bit in this register sets the corresponding bit in the Socket Event Register and / or the Present State Register.

©2001 PCMCIA/JEITA

Default Value	Bit #	Description
00000000H	0	*CSTSCHG* - Sets the Card Status Change bit in the Event Register. The Present State Register remains unchanged.
	1-2	*Card Detect Changed* - Sets the Card Status Change bit in the Event Register. The Present State Register remains unchanged.
	3	*Power Up* - Sets the PowerCycle bit in the Event Register. The Present State Register remains unchanged.
	4	*16-bit PC Card* - Sets the *16-bit PC Card* bit in the Present State Register. If a card is installed in the socket, writes to this bit are ignored.
	5	*CBCard* - Sets the *CBCard* bit in the Present State Register. If a card is installed in the socket, writes to this bit are ignored.
	6	Reserved
	7	*NotACard* - Sets the *NotACard* bit in the Present State Register. If a card is installed in the socket, writes to this bit are ignored.
	8	*Data Lost* - This bit will cause the Data Lost bit to be set in the Present State Register.
	9	*BadVccReq* - This bit will cause the *BadVccReq* bit in the Present State Register to be set.
	10	*5VCard* - This bit will cause the *5VCard* bit in the Present State Register to be set. Writes to this bit disables the bridge's ability to power up the socket. To change the voltage of a card, after forcing this bit, the bridge must either receive a **PCIRST#** or retest the card's supported voltages. The latter can be accomplished by forcing the *CVSTest* bit. This is necessary to prevent software from applying an incorrect voltage to the bridge.
	11	*3VCard* - This bit will cause the *3VCard* bit in the Present State Register to be set. Writes to this bit disables the bridge's ability to power up the socket. To change the voltage of a card, after forcing this bit, the bridge must either receive a **PCIRST#** or retest the card's supported voltages. The latter can be accomplished by forcing the *CV Test* bit. This is necessary to prevent software from applying an incorrect voltage to the bridge.
	12-13	Reserved
	14	*CV Test* - Causes the controller to test the **VS** and **CCD** lines to determine card type and voltages supported. This test is run automatically when a new card is inserted.
	15-26	Reserved - This bit returns zero (0) when read.
	27	*FZVSupport* – Force Zoomed Video Support. Writes to this bit cause the ZVSUPPORT bit in the socket present state register to be written.
	28-31	Reserved

4.5.3.5 Control Register (Offset = 10H)

The Socket Control Register provides control of the socket's Vcc and Vpp/Vcore. All bits in this register should be set to zero by **PCIRST#** and power removed from the socket.

<table>
<tr><th>Default Value</th><th>Bit #</th><th>Description</th></tr>
<tr><td>00000000H</td><td>0-2</td><td>Vpp/Vcore Control - Used to switch the Vpp power using external Vpp/Vcore control logic. The bridge has no knowledge of a card's Vpp/Vcore voltage requirement. Software must determine the needed voltage from the card's CIS.
<table>
<tr><th>Bit #2</th><th>Bit #1</th><th>Bit #0</th><th>Vpp/Vcore Requested</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0V</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>12.0V</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>5.0V</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3.3V</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1.8V</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td></td><td>3</td><td>Unused - Returns zero when read.</td></tr>
<tr><td></td><td>4-6</td><td>Vcc Control - Used to control the power to the PC Card via external control logic. The bridge determines the voltages that can be applied by decoding the CD and VS signals per the CardBus chapter of the Electrical Specification. Those bits and the voltages available in the system determine the correct Vcc options. The value written to this register must agree with the value needed to apply the correct value of Vcc. The bridge must not allow an incorrect Vcc voltage to be applied to a socket. The voltages available are shown in the Status Register.
<table>
<tr><th>Bit #6</th><th>Bit #5</th><th>Bit #4</th><th>Vcc Requested</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0V</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>5.0V</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3.3V</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td></td><td>7</td><td>StopClock - When set causes the bridge to stop the CardBus clock (CCLK) using the CLKRUN# protocol. This allows software control of the CLKRUN# protocol in those systems that do not support CLKRUN# on the host side of the controller. Default is 0.</td></tr>
<tr><td></td><td>8</td><td>Reserved – Returns zero (0) when read</td></tr>
<tr><td></td><td>9</td><td>ZVEN – Zoomed Video Enable. This bit enables Zoomed Video for the socket</td></tr>
<tr><td></td><td>10</td><td>STANDARDZVREG – Standardized Zoomed Video register model supported</td></tr>
<tr><td></td><td>11</td><td>ZV_Activity – This bit returns zero (0) when the ZVEN bits (bit 9) for BOTH sockets are 0 (disabled). If either ZVEN bit is set to 1, the ZV_ACTIVITY bit will return 1.</td></tr>
<tr><td></td><td>12-31</td><td>Unused - Returns zero when read</td></tr>
</table>

©2001 PCMCIA/JEITA

4.6 CardBus Interface

The bridge implements both a master and slave interface on CardBus. The bridge CardBus master and slave interfaces must be independent to avoid deadlocks in PCI systems.

4.6.1 Arbitration

The bridge provides arbitration for each socket. The bridge interface has the highest priority on CardBus. Arbitration must be parked at the bridge when the bus is idle.

4.6.2 Addresses

The bridge must compare all addresses received from a card to the other secondary bus's windows. If the address does not reside in either card's space it is presumed to reside on the primary bus.

4.6.3 Commands

The CardBus interface implements a subset of the PCI bus command set.

Interrupt Acknowledge	This command is ignored by the bridge
Special Cycle	The bridge does not generate or respond to Special Cycles.
I/O Read	Implemented per the CardBus specification. If an I/O Read command is detected that is targeted through the bridge the access will be accepted.
I/O Write	Implemented per the CardBus specification.
Memory Read	Will read a single DWord from Memory Mapped I/O address space and will prefetch from addresses in the Prefetchable Memory Address Window.
Memory Write	Implemented per the CardBus specification.
Configuration Read and Write	The bridge will generate a Type 0 Configuration Read or Write Cycle on a secondary bus, when it receives a Type 1 configuration cycle on its primary bus targeted to the secondary bus. The CardBus interfaces on the bridge will not respond to configuration cycles.
Memory Read Multiple	The bridge aliases this command to Memory Read.
Memory Read Line	The bridge aliases this command to Memory Read.
Memory Write and Invalidate	The bridge treats this command the same as a Memory Write.

4.7 Socket Management

4.7.1 Insertion

The CardBus chapter of the ***Electrical Specification*** requires that applying power to a cold socket be software controlled. Despite allowing the software to control when power is applied to a cold socket, the bridge must not allow an incorrect Vcc level to be applied to a card. The bridge has no knowledge of the card's requirements for Vpp / Vcore voltages, if any. This places the burden, of only applying the correct Vpp / Vcore, on software. Unfortunately, this makes it possible for a coding error to damage hardware, but it seems to be unavoidable.

Following is a description of the steps followed by the bridge when it first detects a card.

1. Card Detects, VS1 and VS2 are debounced by the bridge.
2. Determine the type of card installed and the Vcc voltage required via the VSn and CDn pins. A state machine for this is described in the CardBus specification.
3. Set the appropriate status bits in the Socket Present State and Socket Event registers. If the card installed is not recognized, the bridge must set the Unsupported Card bit in the Socket Status register.
4. Generate an interrupt if enabled via the Interrupt Mask register.

Socket Services should then poll the Event and State registers on the bridge to determine the cause of the interrupt. Software should then write to the Socket Control register to cause the bridge to apply power to the socket.

After the system writes to the Socket Control register to apply power to the socket the bridge:

1. If the voltage requested by software is either unsupported by the bridge or inappropriate for the card, the bridge will set the BadVccReq bit in the Present State Register and skip to step 3. If the voltage requested is appropriate the bridge performs an external power control cycle.
2. Wait to allow Vcc to stabilize.
3. Generates a PowerCycle interrupt, if enabled.

Until the above cycle is complete the bridge will not allow any accesses to the new card. The new card is only be available via configuration cycles until the necessary memory and/or I/O windows in the card's configuration space are initialized.

Accesses associated with the other socket, if a card is installed, should be allowed to continue during this installation process.

The state machines needed to debounce signals and to determine the card type require a running clock. This means that a card may not be detected in a socket if the clock to the bridge is stopped. In the case of a stopped clock the bridge could ignore a new card until its clock is restarted.

4.7.2 Removal

When a CardBus card is removed from a powered up socket and no transaction associated with that bus is active the bridge will:

1. Assert **CRST#** on the CardBus.
2. Power down the socket.
3. Set the *Card Detect* bit in the Status Event register. The *Card Detect* bits, in the Socket Present State Register, will reflect their current status.
4. Generate an interrupt based on the event, if enabled by the Interrupt Mask.
5. Respond to subsequent accesses to the address mappings for that socket in the manner proscribed by the *Master Abort Mode* bit in the Bridge Control Register.

If a transaction involving the removed card was in process, the bridge will:

1. If the bridge is the master of a CardBus transaction, disconnect from any bus talking to the card in the manner proscribed by the *Master Abort* bit in the Bridge Control Register. If the card was the master the bridge should immediately terminate the transfer on the target bus via a normal disconnect.
2. Assert **CRST#** on the CardBus.

©2001 PCMCIA/JEITA

3. Power down the socket.
4. If any data remains in the bridge's buffers it will be discarded and the *Data Lost* bit in the Socket Event Register will be set.
5. Set the *Card Detect* bit in the Socket Event Register. The *Card Detect* bits, in the Socket Present State Register, will reflect their current status.
6. Generate an interrupt based on the event, if enabled by the Interrupt Mask.
7. Respond to subsequent accesses to the address mappings for that socket in the manner proscribed by the *Master Abort Mode* bit in the Bridge Control Register.

Address mappings in the bridge's registers for that socket will remain until changed by software. This allows a predetermined failure mode when accesses to the departed card are attempted.

4.7.3 Wakeup

If while a socket is powered down, the bridge receives a WAKEUP signal, and the Interrupt mask bit enabling it is set, the bridge will generate an interrupt. It is the responsibility of software to power up a card. The bridge must not automatically provide it on seeing a WAKEUP signal.

5. SOCKET PHYSICAL SPECIFICATION

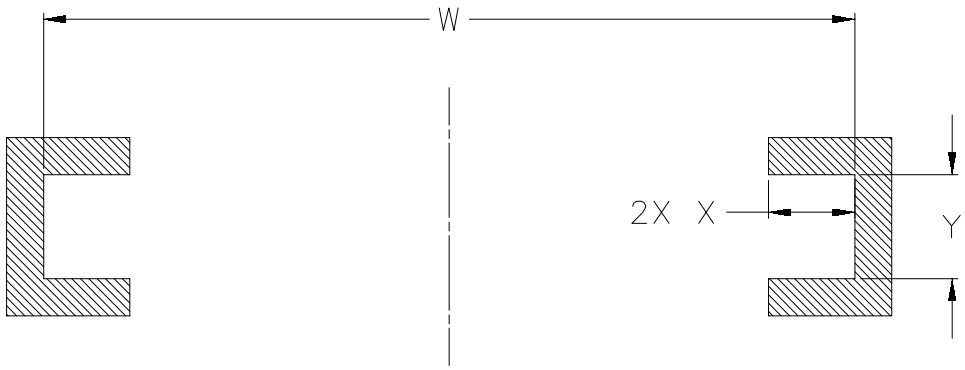
5.1 Scope

This section defines the physical characteristics of the host PC Card socket, necessary to ensure physical compatibility with all PC Card package types. The specifications put forth in this section describe a standard socket, which universally accepts all three full size PC Card package types: Type I, Type II, and Type III

5.2 Goal of this Specification

The goal of this specification is to establish a standard around a universal socket guiderail, so that socket guiderails / ejector assemblies can be used in all applications, and won't exclude certain package types (like Type III). These dimensions must be met to be standard compliant.

5.3 Host Socket Guiderail Dimensions



W MIN	X MAX	Y MIN
54.1	1.5	3.4

Figure 5-1: Host Socket Guide Rail Dimensions

This page intentionally left blank

©2001 PCMCIA/JEITA

射 频 和 天 线 设 计 培 训 课 程 推 荐

易迪拓培训(www.edatop.com)由数名来自于研发第一线的资深工程师发起成立，致力并专注于微波、射频、天线设计研发人才的培养；我们于 2006 年整合合并微波 EDA 网(www.mweda.com)，现已发展成为国内最大的微波射频和天线设计人才培养基地，成功推出多套微波射频以及天线设计经典培训课程和 ADS、HFSS 等专业软件使用培训课程，广受客户好评；并先后与人民邮电出版社、电子工业出版社合作出版了多本专业图书，帮助数万名工程师提升了专业技术能力。客户遍布中兴通讯、研通高频、埃威航电、国人通信等多家国内知名公司，以及台湾工业技术研究院、永业科技、全一电子等多家台湾地区企业。

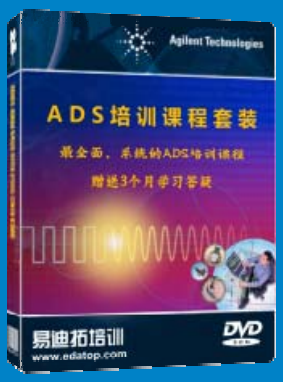
易迪拓培训课程列表：http://www.edatop.com/peixun/rfe/129.html

射频工程师养成培训课程套装

该套装精选了射频专业基础培训课程、射频仿真设计培训课程和射频电路测量培训课程三个类别共 30 门视频培训课程和 3 本图书教材；旨在引领学员全面学习一个射频工程师需要熟悉、理解和掌握的专业知识和研发设计能力。通过套装的学习，能够让学员完全达到和胜任一个合格的射频工程师的要求…

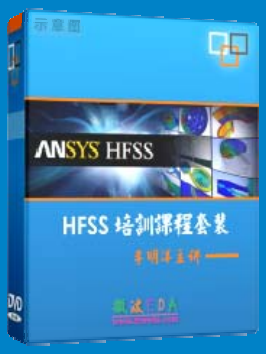
课程网址：http://www.edatop.com/peixun/rfe/110.html

ADS 学习培训课程套装



该套装是迄今国内最全面、最权威的 ADS 培训教程，共包含 10 门 ADS 学习培训课程。课程是由具有多年 ADS 使用经验的微波射频与通信系统设计领域资深专家讲解，并多结合设计实例，由浅入深、详细而又全面地讲解了 ADS 在微波射频电路设计、通信系统设计和电磁仿真设计方面的内容。能让您在最短的时间内学会使用 ADS，迅速提升个人技术能力，把 ADS 真正应用到实际研发工作中去，成为 ADS 设计专家...

课程网址： http://www.edatop.com/peixun/ads/13.html



HFSS 学习培训课程套装

该套课程套装包含了本站全部 HFSS 培训课程，是迄今国内最全面、最专业的 HFSS 培训教程套装，可以帮助您从零开始，全面深入学习 HFSS 的各项功能和在多个方面的工程应用。购买套装，更可超值赠送 3 个月免费学习答疑，随时解答您学习过程中遇到的棘手问题，让您的 HFSS 学习更加轻松顺畅…

课程网址：http://www.edatop.com/peixun/hfss/11.html

CST 学习培训课程套装

该培训套装由易迪拓培训联合微波 EDA 网共同推出，是最全面、系统、专业的 CST 微波工作室培训课程套装，所有课程都由经验丰富的专家授课，视频教学，可以帮助您从零开始，全面系统地学习 CST 微波工作的各项功能及其在微波射频、天线设计等领域的设计应用。且购买该套装，还可超值赠送 3 个月免费学习答疑…

课程网址：http://www.edatop.com/peixun/cst/24.html

HFSS 天线设计培训课程套装

套装包含 6 门视频课程和 1 本图书，课程从基础讲起，内容由浅入深，理论介绍和实际操作讲解相结合，全面系统的讲解了 HFSS 天线设计的全过程。是国内最全面、最专业的 HFSS 天线设计课程，可以帮助您快速学习掌握如何使用 HFSS 设计天线，让天线设计不再难…

课程网址：http://www.edatop.com/peixun/hfss/122.html

13.56MHz NFC/RFID 线圈天线设计培训课程套装

套装包含 4 门视频培训课程，培训将 13.56MHz 线圈天线设计原理和仿真设计实践相结合，全面系统地讲解了 13.56MHz 线圈天线的工作原理、设计方法、设计考量以及使用 HFSS 和 CST 仿真分析线圈天线的具体操作，同时还介绍了 13.56MHz 线圈天线匹配电路的设计和调试。通过该套课程的学习，可以帮助您快速学习掌握 13.56MHz 线圈天线及其匹配电路的原理、设计和调试…

详情浏览：http://www.edatop.com/peixun/antenna/116.html

我们的课程优势：

※ 成立于 2004 年，10 多年丰富的行业经验，

※ 一直致力并专注于微波射频和天线设计工程师的培养，更了解该行业对人才的要求

※ 经验丰富的一线资深工程师讲授，结合实际工程案例，直观、实用、易学

联系我们：

※ 易迪拓培训官网：http://www.edatop.com

※ 微波 EDA 网：http://www.mweda.com

※ 官方淘宝店：http://shop36920890.taobao.com