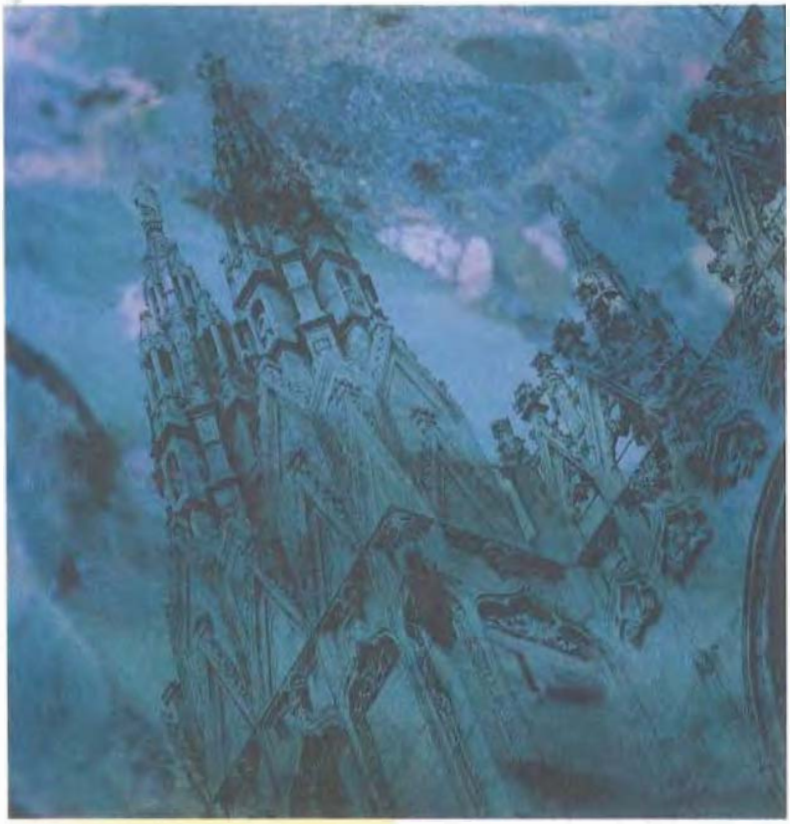
"The 'must-have' PC architecture reference set."

—*PC Magazine's* "Read Only" column

PCMCIA System Architecture



16-Bit PC Cards

Second Edition

MindShare, Inc.

Don Anderson

PC System Architecture Series

PCMCIA System Architecture

16-Bit PC Cards

Second Edition

MINDSHARE, INC.
DON ANDERSON

ADDISON–WESLEY

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data

Anderson, Don, 1953–
PCMCIA system architecture : 16-bit cards / MindShare, Inc., Don Anderson.
p. cm.
Includes index.
ISBN 0-201-40991-7 (alk. paper)
1. PCMCIA cards (Microcomputers) 2. Computer architecture.
I. MindShare, Inc. II. Title.
TK7895.P38A63 1995
004.6'4--dc20 95-44074
CIP

Sponsoring Editor: Keith Wollman
Production Coordinator: Deborah McKenna
Cover design: Barbara T. Atkinson
Set in 10 point Palatino by MindShare, Inc.

Text printed on recycled and acid-free paper.

ISBN 0201409917

7 8 9 101112 MA 06 05 04 03

7th Printing May 2003

The publisher offers discounts on this book when ordered in quantity for special sales. For more information, please contact: Pearson Education Corporate Sales Division, One Lake Street, Upper Saddle River, NJ 07458, (800) 382-3419, corpsales@pearsontechgroup.com

For Doris and Darrel Anderson, my mother and father.

The PC System Architecture Series

MindShare, Inc.

Please see our web site (http://www.awprofessional.com/series/mindshare) for more information on these titles.

AGP System Architecture: Second Edition
0-201-70069-7

CardBus System Architecture
0-201-40997-6

FireWire® System Architecture: Second Edition
0-201-48535-4

InfiniBand System Architecture
0-321-11765-4

ISA System Architecture: Third Edition
0-201-40996-8

PCI System Architecture: Fourth Edition
0-201-30974-2

PCI-X System Architecture
0-201-72682-3

PCMCIA System Architecture: Second Edition
0-201-40991-7

Pentium® Pro and Pentium® II System Architecture: Second Edition
0-201-30973-4

Pentium® Processor System Architecture: Second Edition
0-201-40992-5

Plug and Play System Architecture
0-201-41013-3

Protected Mode Software Architecture
0-201-55447-X

Universal Serial Bus System Architecture: Second Edition
0-201-30975-0

HyperTransport™ System Architecture
0-321-16845-3

Contents

About This Book

Part One
Introduction to PCMCIA

Chapter 1: The Problem

Chapter 2: The PCMCIA Solution

Chapter 3: Tying the Pieces Together

Part Two
Socket and Host Bus Adapter Design

Chapter 4: The Physical Specifications

Chapter 5: The Memory-Only Socket Interface

Chapter 6: The Memory or I/O Interface

Chapter 7: The DMA Interface

Chapter 8: The ATA Interface

Chapter 9: The AIMS Interface

Chapter 10: The PC Card Host Bus Adapter

Part Three
PC Card Design

Chapter 11: The Card Information Structure (CIS)

Chapter 12: Function Configuration Registers

Chapter 13: An SRAM Card Example

Chapter 14: A Flash Card Example

Chapter 15: A FAX/Modem Example

Chapter 16: An ATA PC Card Example

Chapter 17: A Multiple Function PC Card Example

Part Four
PCMCIA Software

Chapter 18: The Configuration Process

Chapter 19: Socket Services

Chapter 20: Card Services

Chapter 21: Client Drivers

Chapter 22: Booting from PC Cards

Chapter 23: Execute In Place (XIP)

Part Five
ExCA (QuickSwap)

Chapter 24: ExCA (QuickSwap)

Part Six
An Example HBA

Chapter 25: An Example HBA—The CL-PD6722

Appendices

Figures

Tables

Tables

Special Recognition

Special thanks to Tom Shanley, my best friend, business partner and hiking companion, who keeps me on the right path.

Acknowledgments

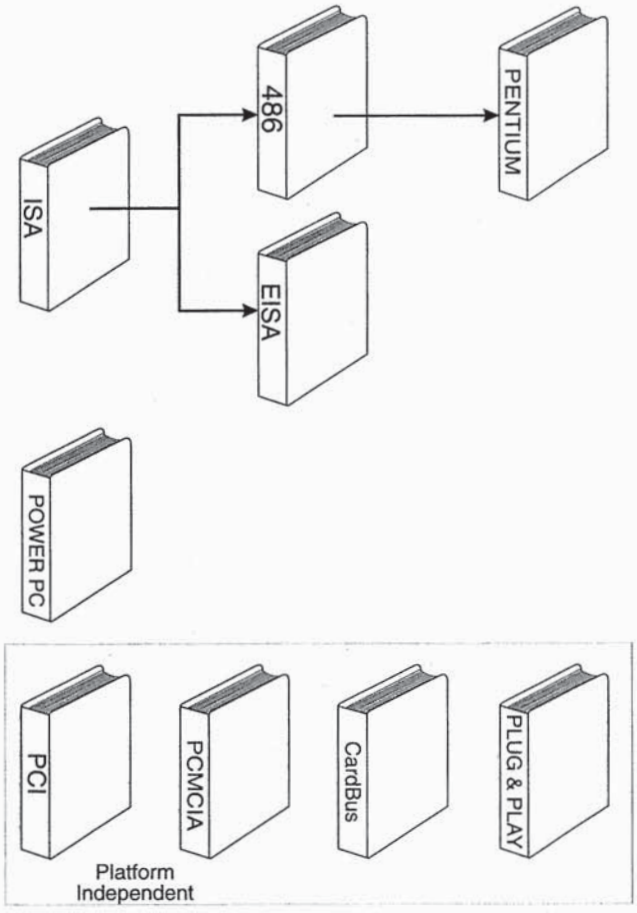
I extend my appreciation and gratitude to the developers at IBM and Intel who provided valuable information and insight during the development of this book and training course. Special thanks to those at the IBM Toronto site who struggled with me during the early stages. Thanks also to those at Cirrus Logic who answered many questions and provided valuable information. Finally, I would like to thank Maxtor for providing information on their ATA drive.

Thanks to those at Norand in Cedar Rapids for their efforts in catching many errors in the manuscript and suggesting improvements to both text and illustrations.

The MindShare Architecture Series

The series of books by MindShare on system architecture includes; *ISA System Architecture, EISA System Architecture, 80486 System Architecture, PCI System Architecture, Pentium™ Processor System Architecture, PCMCIA System Architecture, PowerPC™ System Architecture, Plug and Play System Architecture,* and *CardBus System Architecture,* all published by Addison-Wesley.

Rather than duplicating common information in each book, the series uses a building-block approach. *ISA System Architecture* is the core book upon which the others build. The figure below illustrates the relationship of the books to each other.



Architecture Series Organization

Organization of This Book

PCMCIA System Architecture is organized into six parts consisting of twenty-five chapters. A brief description of each chapter follows:

Part One — Introduction to PCMCIA

Chapter 1: The Problem. This chapter focuses on the industry needs that led to the emergence of PCMCIA and the development of PC Card Standard.

Chapter 2: The PCMCIA Solution. This chapter discusses the emergence of PCMCIA, traces its evolution and introduces terminology and key concepts behind PCMCIA. Key features of the latest release of the 16-bit PCMCIA standard (called the PC Card Standard) are also introduced.

Chapter 3: Tying the Pieces Together. This chapter explains the relationships between the various hardware and software elements employed in a typical PC Card environment. The elements discussed include: the PCMCIA Host Bus Adapter (HBA); the PC Card socket; the PC Card; socket services; card services; and enablers.

Part Two — Socket and Host Bus Adapter Design

Chapter 4: The Physical Specifications. This chapter focuses on the various physical packages defined by PCMCIA for PC Cards and the related environmental specifications. The chapter also describes the standard socket and low-voltage socket types.

Chapter 5: The Memory-Only Socket Interface. This chapter details the memory-only electrical interface between the PC Card and socket. Each pin is defined and it relationship to the PC Card and the HBA is discussed. The memory-only interface is the interface initially seen by 16-bit PC Cards when they are first inserted into a socket. This permits the memory-mapped CIS to be accessed to determine the PC Card type and interface requirements. If the card is designed for an interface type other than memory-only, then the HBA and PC Card are configured to communicate via one of the other interfaces defined by the PC Card Standard (discussed in the following chapters). Also

discusses the timing of socket accesses to PC Cards of differing speeds, including transfers with attribute memory and common memory.

Chapter 6: The Memory or I/O Interface. This chapter details the memory or I/O socket interface. The memory-only interface is converted into a memory or I/O interface by software after it detects that an I/O PC Card has been installed into the socket interface. Pins that are added or redefined by the memory or I/O interface are discussed along with their relationship to the I/O card function. Some of the memory-only pins are replaced with I/O specific pins when the interface is redefined for I/O. This chapter describes how the functions associated with the replaced memory-only pins are handled.

Chapter 7: The DMA Interface. This chapter defines the DMA compliant electrical interface, permitting I/O Cards to use PC compatible DMA transfers. The DMA interface allows I/O devices that use DMA to take advantage of existing compatible software when performing data transfers.

Chapter 8: The ATA Interface. This chapter discusses the PC Card ATA interface. An PC Card ATA interface provides a PC compatible hardware and programming interface that simplifies the job of implemented hard drive solutions in the PCMCIA environment. This chapter defines the various ways that a PC Card ATA can be mapped in the system along with the electrical interfaces that are used. Differences between the PC compatible ATA implementation versus the PC Card ATA interface are also discussed.

Chapter 9: The AIMS Interface. This chapter focuses on the optional Auto-Indexing Mass Storage (AIMS) interface. The transfer mechanism is described, along with the registers that must be programmed to initiate the transfer.

Chapter 10: The PC Card Host Bus Adapter. This chapter discusses the role of the PCMCIA Host Bus Adapter. Individual Host Bus Adapter functions are discussed. A functional block diagram of an HBA adapter is provided along with detailed explanations of each function.

Part Three — PC Card Design

Chapter 11: The Card Information Structure (CIS). This chapter discusses layer one of the metaformat, commonly referred to as the card information structure, or CIS. The chapter details the role of the CIS in the PC Card configuration process. Tuples are also introduced and their format and structure

are described. The basic structure of the CIS's configuration table required by I/O cards is also described.

Chapter 12: Function Configuration Registers. This chapter discusses the configuration registers and provides a complete description of each register specified by the PC Card standard. Configuration register implementations for both single and multiple function cards are covered.

Chapter 13: An SRAM Card Example. This chapter describes a sample SRAM card implementation, including a functional block diagram of the SRAM card along with a sample CIS.

Chapter 14: A Flash Card Example. This chapter describes a flash card implementation, including a functional block diagram of the card, a sample CIS, and configuration registers implemented by the card.

Chapter 15: A FAX/Modem Example. This chapter describes an example FAX/Modem implementation, including a functional block diagram, sample CIS, and related configuration registers.

Chapter 16: An ATA PC Card Example. This chapter describes an example PC Card ATA drive implementation, including a functional block diagram, a sample CIS, and configuration registers implemented by the card.

Chapter 17: A Multiple Function PC Card Example. This chapter discusses the multiple function PC Card strategy and the mechanisms for achieving it. It also includes a functional block diagram of a multiple function PC Card, a sample multi-function CIS, related configuration registers, and multi-function interrupt handling.

Part Four — PCMCIA Software

Chapter 18: The Configuration Process. This chapter provides an overview of the PCMCIA software environment and the configuration process. The primary role and interaction between each piece of software is established. This chapter also introduces the common software solutions provided along with the most popular suppliers.

Chapter 19: Socket Services. This chapter discusses the role of socket services. It also describes the initialization of socket services and explains the basic purpose of the functions commonly supported in the PC environment.

Chapter 20: Card Services. This chapter focuses on the role of card services in the PCMCIA environment. This chapter review each of the functions defined by the PC Card specification that apply to 16-bit PC Cards, along with related return codes. The call back mechanism is also described and the event and call back codes are defined.

Chapter 21: Client Drivers This chapter discusses the three basic types of enablers: point enablers, device-specific enablers, and super enablers. The chapter also discusses the jobs performed by generic memory enablers (and MTDs) and I/O device enablers.

Chapter 22: Booting from PC Cards. This chapter discusses the problems associated with loading the operating system from a PC Card. It also defines mechanisms used to determine whether a given PC Card is a bootable device, and the firmware support required to support PC Card booting.

Chapter 23: Execute In Place (XIP). This chapter discusses the Execute-In-Place mechanism defined by PCMCIA that allows code to be executed directly from the card rather than copying files to and executing from system memory.

Part Five — ExCA (QuickSwap)

Chapter 24: ExCA (QuickSwap). This chapter introduces the ExCA (QuickSwap) specification that defines a required set of hardware and software support, intended to improve PC Card interoperability across platforms based on Intel x86 architecture.

Part Six — An Example HBA

Chapter 25: An Example HBA–The CL-PD6722. This chapter provides an overview of a sample PCMCIA host bus adapter (The Cirrus Logic CL-PD6722) used in Intel x86 implementations for either an original PC or ISA compatible host bus.

Appendices

SRAM CIS Example

Flash CIS Example

FAX/Modem CIS Example

ATA Disk CIS Example

Metaformat Layers 2, 3, and 4

References

Who Should Read This Book

This book is intended for use by hardware and software designers and support personnel. Due to the clear and concise explanatory methods used to describe each subject, personnel outside of the design field may also find the text useful.

Prerequisite Knowledge

We highly recommend that you have a thorough knowledge of PCs, including hardware and software interaction prior to reading this book. Several MindShare publications provide all of the background necessary for a complete understanding of the subject matter covered in this book. Much of the background information can be obtained from the *ISA System Architecture* book.

Documentation Conventions

This section defines the typographical conventions used throughout this book.

Hex Notation

All hex numbers are followed by an "h". Examples:

9A4Eh

0100h

Binary Notation

All binary numbers are followed by a "b". Examples:

0001 0101b

01b

Decimal Notation

When required for clarity, decimal numbers are followed by a "d". Examples:

256d

128d

Signal Name Representation

Each signal that assumes the logic low state when asserted is followed by a pound sign (#). As an example, a PC Card modem asserts the IREQ# signal to a logic low state when signaling an interrupt request to the system.

Signals that are not followed by a pound sign are asserted when they assume the logic high state. As an example, a PCMCIA Card asserts READY to logic high state, indicating that it is ready to be accessed.

Identification of Bit Fields (logical groups of bits or signals)

All bit fields are designated as follows:

[X:Y],

where "X" is the most-significant bit and "Y" is the least-significant bit of the field. As an example, the PCMCIA socket supports address lines A[25:0], where A25 is the most-significant and A0 the least-significant bit of the address.

CardBus

An enhanced version of PCMCIA is also defined by the PC Card standard. The new high-speed CardBus cards incorporate 32-bit data transfers and bus mastering capability. See MindShare's *CardBus System Architecture* book published by Addison-Wesley for details regarding the CardBus implementation.

We Want Your Feedback

MindShare values your comments and suggestions. You can contact us via mail, phone, fax, or internet email.

E-Mail/Phone/FAX

Email: mindshar@interserv.com

Phone: (214) 231-2216

Fax: (214) 783-4715

Mailing Address

Our mailing address is:

MindShare, Inc.

2202 Buttercup Drive

Richardson, Texas 75082

Part One

Introduction to PCMCIA

Chapter 1

This Chapter

This chapter focuses on the industry needs that led to the emergence of PCMCIA and the development of PC Card Standard.

The Next Chapter

The next chapter introduces the PCMCIA solution and reviews the evolution of the PCMCIA Standard.

The Mobile Computing Environment

The growth of the microcomputer industry in the 1980s and the popularity of the PC led to the proliferation of laptop, notebook and sub-notebook computers. Manufacturers strived to deliver desktop performance in smaller and lighter portable systems, powered by batteries. This fueled the need for lighter, smaller, and less power hungry peripheral devices. A major focus of this effort revolved around the relatively large, heavy, power hungry floppy drive subsystem.

In addition to being small, lighter, and more power efficient, the alternative system had to provide many of the same characteristics of the floppy disk; it had to include removable media that was transportable to other systems, and had to be immediately accessible when installed into the system for reading and writing files. Early interest revolved primarily around the use of battery-backed memory cards implemented as a virtual floppy drive subsystem. Memory cards were physically small, could store large amounts of data, and consumed relatively little system power when compared to the floppy drive. Furthermore, the emergence of Flash memory promised to provide an economical memory card solution that required no battery back-up.

Small Form-Factor I/O Expansion Devices

The mobile computer environment also had a need for small and power efficient I/O expansion devices. The small PCMCIA form-factor drew attention as a possible solution for I/O expansion devices. The initial PCMCIA designs supported only memory cards however, the need to expand PCMCIA to include I/O device support was clear.

Chapter 2

The Previous Chapter

The previous chapter focused on the industry needs that led to the emergence of PCMCIA and the development of PC Card Standard.

This Chapter

This chapter discusses the emergence of PCMCIA, traces its evolution and introduces terminology and key concepts behind PCMCIA. Key features of the latest release of the 16-bit PCMCIA standard (called the PC Card Standard) are also introduced.

The Next Chapter

The next chapter explains the relationships between the various hardware and software elements employed in a typical PC Card environment. The elements discussed include: the PCMCIA Host Bus Adapter (HBA); the PC Card socket; the PC Card; socket services; card services; and client drivers.

The Virtual Floppy Drive Subsystem

Solid state memory cards can provide an alternative to the mechanical floppy and floppy drive. In other words, memory cards can be implemented as a virtual floppy drive subsystem. Such a solution must permit standard PC software to access the memory cards as if they were floppy disks. This necessitates translation of PC compatible software calls used to access an ordinary floppy disk into commands that access files stored within the memory card. To ensure compatibility with existing PC software a standardized software protocol was required to ensure compatible operation of the memory cards.

Memory cards implemented as virtual floppy disks must also have the ability to be inserted and removed from the system at any time as is done with

floppy disks. When a card is installed, software must be able to access the files stored on the memory card or write new files to it. How was this to be done? Several key questions come to mind: How would the insertion of a memory card be detected? Were memory cards to be accessed via an I/O port as done with the floppy drive interface, or mapped into the processor's memory address space? What system resources would be required? What software would be responsible for the various aspects of recognizing, configuring, and accessing the memory cards? These questions and others clearly pointed to the need for hardware and software standards that could be implemented by system manufacturers to ensure interoperability of memory cards between IBM compatible systems.

The Lack of a Standard Memory Card Design

Numerous memory card manufacturers produced cards with differing physical and electrical properties, making compatibility a major obstacle in fulfilling industry needs. A standard physical package, electrical interface and connector were needed to ensure compatibility of memory cards.

Emergence of PCMCIA

Several manufacturers met in the summer of 1988 to investigate the possibility of forming a standards organization to deal with memory card standards and interoperability issues. A year later the Personal Computer Memory Card International Association (PCMCIA) was founded, and the first PCMCIA Standard (Release 1.0) was introduced in September 1990. This standard specified the design of memory cards (commonly called PC Cards) and a socket interface to be implemented as virtual disk drives.

PCMCIA was formed to promote the standardization and interchangeability of PC Cards. Initially, its primary focus was defining PC Card standards for IBM PC-compatible (DOS-based) systems. The long-term goal is to allow a variety of computer types and non-computer products to freely interchange PC cards. With these goals in mind the PCMCIA defined standards for PC Cards.

The Japanese Electronics Industry Development Association (JEIDA) began working on memory card standardization issues in 1985. In 1989 PCMCIA adopted JEIDA's 68-pin connector as its socket interface. To serve the goals of

compatibility and interoperability JEIDA and PCMCIA began working jointly to ensure compatibility between their standards. In 1990 PCMCIA announced its first standard (release 1.0) and JEIDA released its fourth standard (release 4.0). As newer versions of the standards are released, JEIDA and PCMCIA continue to work closely to support each other's standards.

Support for I/O-based PC Cards Added

The mobile computing environment also needed standardized small form-factor I/O devices that could be added to mobile systems as expansion devices. PCMCIA's release 2.0 added support for I/O devices that could be inserted into a PCMCIA socket. Like memory cards, these devices are designed to be automatically detected by the system when installed and automatically configured. This gives PC Cards the ability to be inserted into a PCMCIA socket after the system has already been powered up and is operational.

The PC Card Standard

The PCMCIA standard defines the following major items:

- Physical design of the PC Card
- Physical design of the connector (socket)
- Electrical interface to PC Cards
- Software architecture

The PC Card standard has been designed with flexibility in mind, allowing PC Card socket implementations to be adapted for a wide variety of systems. Major features of today's PCMCIA standard include items listed in table 2-1.

PC Cards come in a wide range of memory and I/O devices. Memory devices include RAM, FLASH memory and various types of ROM. I/O devices include voice, data and FAX modems; network interface cards; wireless communications (such as, Global Positioning Systems (GPS), pagers and networks); AT Attachment (ATA) Hard Drives (also called IDE drives); small computer system interface (SCSI) adapters; and many others.

Three sizes of PC Cards are specified by the physical standard. Each type of card has the same electrical interface and planar dimensions, but the thickness

varies to accommodate designs that require more physical space. Generally, type I cards (3.3mm thick) are used for memory devices of various kinds, type II cards (5.0mm thick) for modems, LANs, etc., and type III cards (10.5mm thick) for devices such as ATA hard drives.

Table 2-1. PCMCIA Feature Summary

Feature	Description
Small Form-Factor: Three Physical Device Types Defined	PCMCIA cards have a standard length and width of 85.6mm (3.370") X 54.0mm (2.126"). The card type determines the card's thickness — Type 1 = 3.3mm (0.130"); Type 2 = 5.0mm (0.197"); Type 3 = 10.5mm (0.413").
Host Bus Independence	PCMCIA sockets can be connected to a wide variety of host buses. Sockets are connected to the host systems via host bus adapters designed for a particular bus interface.
Three Address Spaces	PCMCIA supports common memory address space (standard memory addresses), attribute memory space (for automatic configuration) and I/O address space.
64 MB of Address Space	Twenty-six address lines provide address space up to 64 MB for each address space.
16-bit Data Path*	Sixteen data lines permit word transfers to/from PC Cards.
I/O Device Support	I/O devices as well as memory devices can be implemented in the credit card form-factor.
Direct Memory Access (DMA) Support	The PC Card standard incorporated DMA support so that standard PC expansion devices that use DMA can be supported in PC Card implementations and take advantage of the existing software.
Multifunction PC Cards Support	The PC Card standard directly supports PC Cards that include multiple memory or I/O functions or both.
Automatic Configuration	When installed, PC Cards are configured automatically without the need for user intervention.
Software Transparency	Software written for standard host bus devices can be used when accessing the same device that is implemented in a PC Card. Once the PC Card is installed and configured it typically behaves like any other host device.
Easy to Implement Configuration Software	PCMCIA provides a standard software interface, simplifying the design and implementation of device-drivers required to configuration PC Cards.
Low Voltage Support	The PC Card standard supports 5 volt, 3.3 volt and what PCMCIA refers to as X.X voltage (an arbitrary low voltage to be specified sometime in the future).

Table 2-1 PCMCIA Feature Summary (continued)

Feature	Description
Support for Several Different File Systems on a Single Card	PC Cards provide a means for specifying support for a variety of different data-recording formats and data organizations.
Execution of Code Directly from Memory Card	PCMCIA memory cards, typically implemented as virtual disks, can be accessed directly for code, without copying it to main memory. This support requires a software protocol called XIP (execute-in-place).

* The PC Card standard also defines a 32-bit PC Card and socket interface called CardBus.

Summary of PCMCIA Releases

Since the first PCMCIA standard was released, many revisions and enhancements have been made. Table 2-2 highlights the chronology of releases, providing a perspective of the pace of change that has occurred in a relatively short period of time. The most recent release (February 1995) is called the PC Card standard, consisting of a 12 volume set listed in table 2-3.

Table 2-2. Evolution of the PCMCIA Specification

Specification	Version	Release Dates
Card Standard	1.0	November, 1990
	1.01	September, 1991
	2.0	November, 1992
	2.1	July, 1993
Socket Services	A.0	June, 1991
	1.00	August, 1991
	1.01	September, 1991
	2.0	November, 1992
	2.1	July, 1993
Card Services	1.0 (draft)	December, 1991
	2.0	November, 1992
	2.1	July, 1993
ATA Interface	1.0	July, 1992
	1.01	November, 1992

Table 2-2. Evolution of the PCMCIA Specification (continued)

Specification	Version	Release Dates
Auto-Indexing Mass Storage (AIMS)	1.0 1.01	July, 1992 November, 1992
Card Extensions	1.0	November, 1992

Table 2-3. List of Individual Volumes Included in the PC Card Standard

Volume Name	Description of Contents
Overview and Glossary	This volume introduces each volume and provides a glossary of terms.
Electrical Specification	The electrical specification provides definition of the socket interface pins, signaling environment, and transfer timing and control.
Physical Specification	The physical specification describes the card and socket dimensions, mechanical specifications, and environmental storage and operational parameters that must be met.
Metaformat Specification	This specification describes a four layer model that encompasses the basic compatibility layer that all PC Cards must implement. The compatibility layer describes a variety of PC Card characteristics and capabilities needed to configure the card. The subsequent layers define a memory card's method of recording data and describe its organization. This information provides software with the information it needs to manage access to a variety of PC memory cards in a compatible fashion.
Card Services Specification	This specification defines function calls used by a PC Card's client driver to configure and control access to it's PC Card.
Socket Services Specification	This specification defines function calls used principally by Card Services to access a particular HBA. These functions are comparable to BIOS functions provided in IBM compatible PCs.

Table 2-3. List of Individual Volumes Included in the PC Card Standard (continued)

Volume Name	Description of Contents
Media Storage Formats Specification	This specification describes how data is formatted on many PC Cards that are used as virtual disks. This information can be used to help provide exchangeability of PC memory cards between different host systems.
PC Card ATA Specification	The ATA specification describes the electrical and programming interface required by PC Cards implemented as ATA devices.
XIP Specification	XIP describes the programming interface required by applications that support execution directly from the PC Card file, rather than having to copy the executable file to system memory and executing from there.
Guidelines	This volume provides guidelines for the implementation of a variety of PC Card types.
PCMCIA Extensions	The PCMCIA extensions document features that are not supported by JEIDA, but which are offered as optional capabilities by PCMCIA.
JEIDA Extensions	The JEIDA extension document features that are not supported by PCMCIA, but which are offered as optional capabilities by JEIDA.

Chapter 3

The Previous Chapter

The previous chapter discussed the emergence of PCMCIA, traced its evolution and introduced terminology and the key concepts behind PCMCIA.

This Chapter

This chapter explains the relationships between the various hardware and software elements employed in a typical PC Card environment. The elements discussed include: the PCMCIA Host Bus Adapter (HBA); the PC Card socket; the PC Card; socket services; card services; and enablers.

The Next Chapter

The next chapter focuses on the various physical packages defined by PCMCIA for PC Cards and the related environmental specifications. The chapter also defines the socket interface types.

Overview

This section introduces key PCMCIA terms and discusses the relationship of the major hardware and software elements typically implemented in PCMCIA host system and PC Card designs. Later chapters detail each of these major elements and discuss specific implementations. The PCMCIA solution typically consists of:

Hardware

- The 16-Bit PC Card
- PCMCIA Socket
- The PCMCIA Host Bus Adapter (HBA)

Software

- Socket Services
- Card Services
- PC Card Enablers

The PCMCIA hardware consists of the PC Card, card socket and the host bus adapter (HBA). The HBA bridges the host expansion bus (e.g. ISA, EISA, Micro Channel, or PCI) to the PC Card socket or sockets.

The PCMCIA software architecture consists primarily of a PC Card's enablers, card services and socket services. These software layers exist to support PCMCIA's automatic configuration and "" capability. The term hot insertion is frequently used to refer to the ability of PC Cards to be inserted into a socket when power is applied to the PC. The system automatically detects and configures the PC Card allowing the system to treat it as a floppy disk. That is, once a PC memory card is inserted into a socket, the user can read files or write files just as if a floppy disk had been inserted into a floppy drive. From the user's perspective, the ability to format the memory card (virtual disk), perform chkdsk commands and read and write files is the same as with a floppy drive. The only perceptible difference is the lightning speed at which these operations occur when compared to the speed of the same operations performed with a floppy drive. This is due to the much faster access to solid state media.

Figure 3-1 illustrates the relationships between each of the hardware and software elements that comprise a PCMCIA solution, and identifies the typical supplier of each element. The definition and need for each element is described in the following sections.

The PC Card

A wide variety of hardware applications can be implemented using PC Cards. Many PC Cards consist of memory devices used to emulate floppy or hard drives. These virtual drives provide very fast access and provide a data transfer medium much less susceptible to harsh environmental conditions when compared to magnetic media (such as diskettes and magnetic tape). If these devices are to be used as replaceable media, the system must be able to recognize when a card is inserted or removed and provide the ability to access the card as if it were a floppy disk.

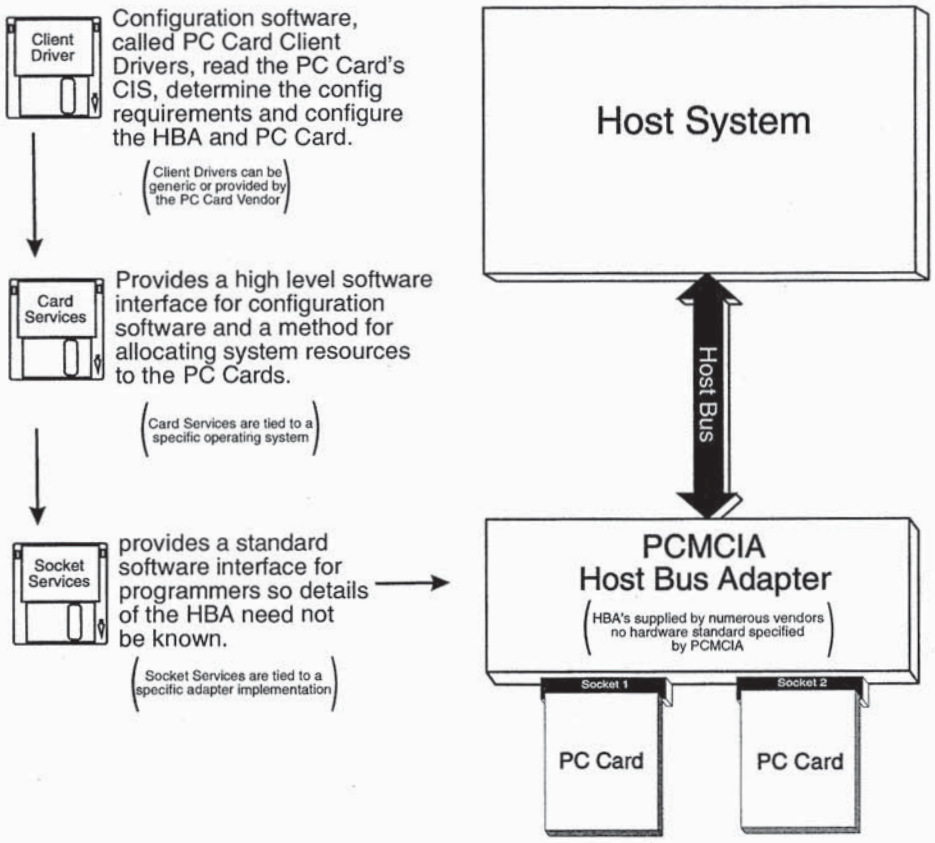


Figure 3-1. Relationship of PCMCIA Software and Hardware

PCMCIA also supports I/O devices such as modems, IDE (ATA) hard drives and LAN controllers. New support has also been added for multifunction devices.

Regardless of the type of PC Card implemented (memory, I/O, or multifunction), PCMCIA systems are designed to permit their automatic detection and configuration. The PCMCIA automatic configuration capability is in some ways similar to that employed by the Micro Channel and EISA system designs, but eliminates the user intervention required with these systems and permits insertion and configuration while power is applied to the system.

In Micro Channel and EISA designs, the manufacturer of an adapter board must supply a configuration file that describes the possible configuration options for the board. The system manufacturer also supplies a configuration file

describing the resources used by the basic system, along with a variety of configuration options. A configuration utility program must be run by the user to merge all the configuration information together (system and installed boards). The configuration software then determines an overall conflict-free configuration that satisfies the requirements of the system and all expansion boards.

PC Cards incorporate configuration option information in non-volatile memory within the card itself, rather than requiring a companion diskette that contains the configuration option information. This configuration data is kept in an area within the card known as the (CIS). The CIS is mapped into the PC Card's attribute memory space. Refer to figure 3-2. PC Card enabler software installs each time the system is powered up. This software checks for the presence of PC Cards that are installed. If a PC Card is detected, the software reads the contents of its CIS to determine the type of device it is, the system resources that it requires, and the configuration options that are possible. The enabler is then responsible for configuring the PC Card so it can be accessed.

PC cards, unlike the Micro Channel and EISA boards, can be inserted either before or after the system is powered up. As a result, enablers must remain available for PC Cards in the event a card is installed following system power-up. Hardware notifies the software when a new card is installed, so that it can be properly configured into the system. When a card is removed, again hardware detects the card's removal and notifies software that the device is no longer installed.

The PCMCIA configuration software consists of one or more PC Card enablers that recognize, enable, and configure the PC Card. This software sometimes comes with a particular PC Card (typically as part of a device driver) that the card manufacturer provides when the PC Card is purchased. This type of PC Card enabler is usually responsible for installing only the PC Card for which it was designed. Other PC Card enablers, sometimes called generic or super-enablers, are typically supplied by the system manufacturer.

Generic enablers evaluate the configuration requirements of a card by scanning and evaluating (parsing) the CIS. If the generic enabler recognizes the card type, it will then attempt to configure the card itself or dynamically load the appropriate enabler from a library residing on disk. Ideally, a single PC Card super-enabler would be used to detect and configure all PC Cards. For a variety of reasons (to be discussed later), a single generic or super-enabler is not always possible today. In short, PCMCIA configuration software may

consist of one or more PC Card enablers designed to identify and configure PC Cards.

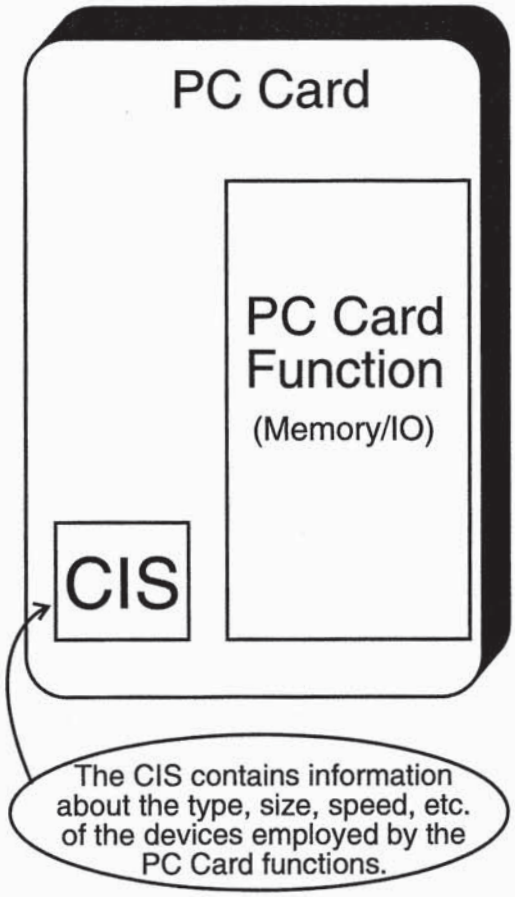


Figure 3-2. The Card Information Structure Contains Configuration Options for the PC Card

Interoperability: PCMCIA Sockets and The PCMCIA Host Bus Adapter

PCMCIA sockets can be designed into a wide variety of PC bus architectures as well as non-PC designs, permitting interoperability of PC Cards. This connection between a given host bus and a PCMCIA socket is provided through a

PCMCIA host bus adapter (HBA). The adapter acts as a bridge, passing host bus transactions to PC Cards installed in PCMCIA sockets. Refer to figure 3-3. Host bus adapters must be programmed by the system in order to gain access to PC Cards. The manner in which the host bus adapter is programmed is determined by the requirements of the particular PC Card installed in the sockets it controls.

Each PCMCIA socket has its own dedicated signal interface provided by the HBA. That is, signals are not bussed between sockets as in most expansion bus architectures. This of course means that the host bus adapter must have a separate interface to each socket along with a single interface to the host system. Access to each socket therefore is controlled through separate sets of socket interface circuitry, each of which must be initialized in order to gain access to a PC Card installed in a given socket.

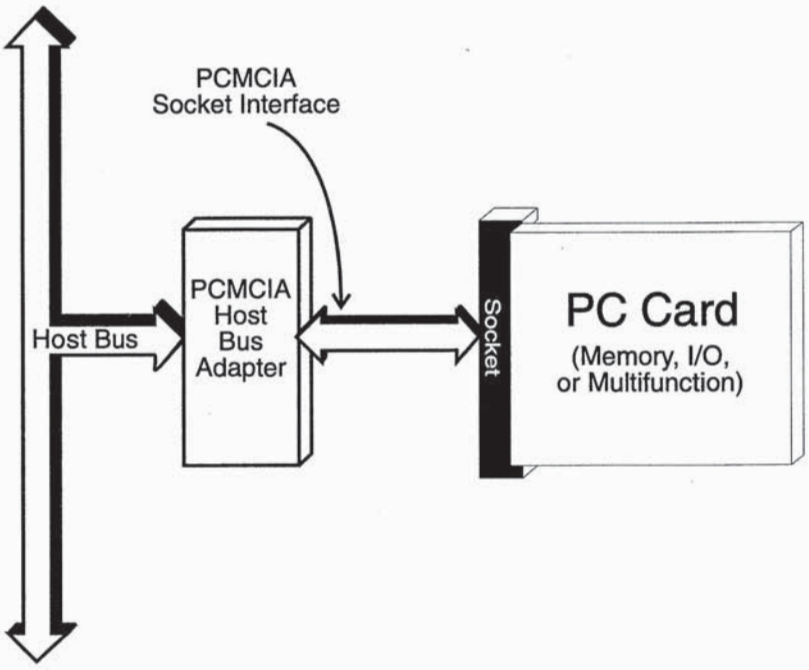


Figure 3-3. PCMCIA Sockets Can Be Incorporated in a Wide Variety of Systems.

Initializing the Host Bus Adapter: Socket Services

Since host bus adapters (HBAs) must have a dual personality — an interface to both the PCMCIA sockets and to the host system — the details of implementing an HBA has been left to the hardware developer and is not included within the PCMCIA specification. This means that a wide variety of HBA designs can (and do) exist for a given type of host interface.

Several chip manufacturers provide PCMCIA host bus adapter chips that system manufacturers can incorporate into their designs. For example, manufacturers of ISA to PCMCIA host bus adapters include: Cirrus Logic, DataBook, Intel, Texas Instruments, Vadem, VLSI Technology, and others. Each of these designs to some extent implement different registers and initialization algorithms. Differences between some HBAs are slight while differences between others are major.

Software must initialize the HBA so that it can pass host expansion transactions to the PC card when it detects an address that resides within the range of addresses used by the PC Card. Since a variety of different HBAs can be used, programmers would have to write their software to interface to each type of host bus adapter in order to ensure proper operation in all system platforms. However, the PC Card specification provides a common software interface called socket services which hides the details of the hardware interface from the programmer. Socket services provides a low-level software interface that gives programmers access to a common set of functions that will permit initialization of any HBA. In IBM PC terminology, these functions can be thought of as merely extensions to the BIOS routines.

Several software vendors have developed socket services for each of the major PCMCIA host adapters. These vendors include: American Megatrends, Award Software, Phoenix Technology, SystemSoft, Ventura Micro Inc., and others. Today most system designers license socket services from a software vendor, making it easier to implement PCMCIA solutions.

Configuring the Card: Card Services & Enablers

PC Cards require configuration software (enablers) that detect their presence, determine their configuration requirements and program them for operation within the system. This is true whether a card is installed when system power is applied or installed after the system is powered up and fully operational.

Configuration software for early PC Cards, built before the introduction of socket services, had to program the host bus adapter to permit access to the card's memory address space. This required that the programmer of the configuration software know the details of the hardware interface and the protocol required to access and program the host bus adapter. Furthermore, these programmers had to poll status registers on the host bus adapter to detect when cards were inserted and removed, or when other status changes occurred at the socket (i.e. low battery detection, write protect switch position, or ready/busy indications). Since different host bus adapters have different hardware interfaces, programmers were required to provide a separate version of their software for each host bus adapter type. The introduction of socket services removed the burden of having to write card enablers to interface directly to the hardware.

In addition, programmers must determine the system resources required by their PC Card and allocate them to the card. The resources that it assigns to its PC Card must not be already allocated to another device in the system. In the past, the programmer had to somehow determine what system resources were available to be allocated, or make an educated guess, hoping that the resources it was about to allocate were not already in use. These issues and others placed a heavy burden on the programmer to ensure that their cards worked in a given system. Fortunately, PCMCIA introduced another software layer called card services that lifts these burdens from the programmer.

Thanks to card services, the job of writing PC Card enablers is a much simpler task today, and far more reliable than was possible with early revisions of the specification. Card services provides a software layer consisting of high-level functions that programmers can call to gain access to a card, determine its configuration requirements, and request the system resources it requires.

One of the primary functions that card services fulfills for enablers is system resource allocation. Card services maintains a data base of system resources that are available for assignment to PC Cards. Once the enabler determines the configuration requirements of the card by reading the card's CIS, it re-

quests that these resources be allocated to its card. If the resource is available, card services returns "success" to the enabler and the resource is then assigned. If the resource has already been used, then card services returns "failure" and another configuration option must be read from the CIS and tried. This process continues until one of the card's configuration options satisfies the request or until all options are exhausted, in which case the card cannot be configured.

Card services also notifies enablers of card insertion and removal and other status change events. This eliminates the need for enablers to continually poll to determine if some status change has occurred.

Accessing PC Cards After Configuration

Once a PC Card has been detected and configured, it behaves like any other device that exists on the expansion (PCMCIA host) bus. This allows applications to access PC Cards directly through the normal methods used in a given operating environment. This access can be done without using the PCMCIA software interface (i.e. card and socket services). Figure 3-4 illustrates the basic software flow during configuration and status reporting, versus run-time access (accesses made to a PC Card by application software once it has been configured). Each of the software components shown in figure 3-4 are discussed in detail in later chapters and the relationships between them are further defined.

The Metaformat

PCMCIA has defined a comprehensive software structure called the metaformat that defines the software support that can be provided with the card. The metaformat is a four layer software model that encompasses the CIS discussed in the previous example. Refer to table 3-1. The only software layer required by all cards is the CIS (layer one), which contains the information necessary to configure the card within the system.

The other layers are intended for memory cards that are used as virtual disk drives. PC Card memory is accessed as a logical disk drive via the operating system's file management system and in conjunction with device drivers that have knowledge of how the PC Card's memory arrays are organized. The additional layers provide information that can be used by file management

software, utilities and other software requiring knowledge of a card's memory array characteristics.

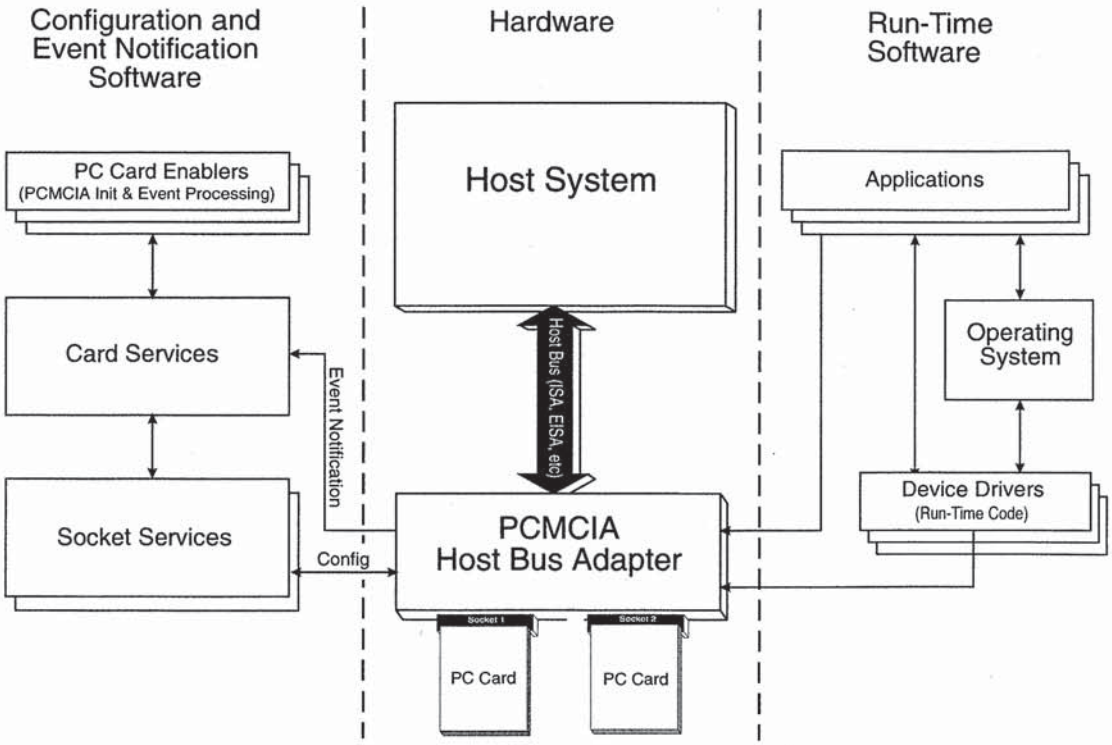


Figure 3-4. Configuration and Status Reporting Software Flow Versus Run Time Software Flow

Few PC Cards today contain information for metaformat layers two through four. In many instances the enablers for memory cards or flash file systems contain information regarding the organization of common devices. The chapter entitled, "The Card Information Structure (CIS)", contains detailed examples of the CIS (layer one of the metaformat) used by typical cards. Refer to Appendix E for a list and description of the tuples defined for layers two through four.

Table 3-1. PCMCIA Card Metaformat

Metaformat Layers	Description
Layer 1 — Compatibility Layer	The compatibility layer includes information necessary for a card to be recognized and configured. This portion of the card standard is commonly referred to as the Card Information Structure (CIS). All 2.0 compliant PC Cards must contain a CIS. The CIS information must be accessible from attribute memory starting at location zero.
Layer 2 — Data Recording Format Layer	This information specifies how data is recorded in the PC Card's memory arrays, and specifies what error checking capabilities are used, if any. Two basic types of data recording are specified: disk-like recording (blocks of data) or memory-like (sequential byte addressable data).
Layer 3 — Data Organization Layer	Defines the logical organization of data within a partition in a memory card. Data within a partition may be organized to support an OS file system, a flash file system, a vendor-specific organization, or an application specific organization.
Layer 4 — System Specific Layer	Specifies application specific information pertaining to a given operating environment. Items currently defined include DOS environment capabilities including: • An interchange format to ensure that PC Cards formatted with a DOS file system can be interchanged with systems implementing all versions of DOS. • Execute in Place (XIP) to support direct execution of application programs from the card. • Ability to configure older cards formatted without a CIS.

Part Two

Socket and Host Bus Adapter Design

Chapter 4

The Previous Chapter

The previous chapter introduced the major concepts of PCMCIA and explained the relationships between the various hardware and software elements employed in a typical PC Card environment.

This Chapter

This chapter focuses on the various physical packages defined by PCMCIA for PC Cards and the related environmental specifications. The chapter also describes the standard socket and low-voltage socket types.

The Next Chapter

The next chapter introduces and defines the memory-only socket interface, including definition of each pin. It also explains and illustrates memory transactions performed to/from PC Card memory functions.

Card Types and Dimensions

Three basic types of physical dimensions are described in the specification for PC Cards. In addition, extensions to type I and type II cards are defined. The card types are:

- Type I
- Type II
- Type III
- Type I extended
- Type II extended

Card Types I, II, and III

Refer to figure 4-1. Type I, II, and III cards all have the same planar or outline dimensions (54.00mm x 85.60mm). Card thickness is also the same within the interconnect area of each card. Only the substrate area of the cards is different. The thickness of each card type in the substrate area is:

- Type I cards = 3.3 mm
- Type II cards = 5.0 mm
- Type III cards = 10.5 mm

Figure 4-1. The Interconnect Area is the Same Thickness for all PC Cards.

Memory cards typically have a write-protect switch. Some memory cards containing volatile RAM devices also include a battery to prevent loss of data when the card is removed from the socket. The PCMCIA specification defines recommended locations for both the battery and write protect switch as shown in figure 4-2.

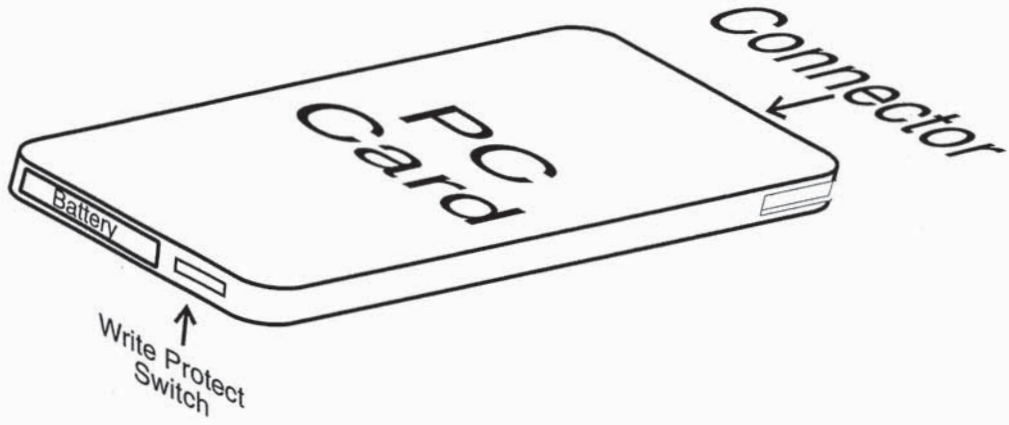


Figure 4-2. Type I Card with Battery and Write Protect Switch

Refer to figure 4-3. Type II cards are most commonly used for I/O devices. The type II package is popular with I/O cards since many of them require more physical space than memory devices due to the state of miniaturization for many of the devices required for I/O designs. These I/O cards typically have an external connector at the end of the card. Previous versions of the PCMCIA standards did not specify the design or location of the external I/O connector. The PC Card standard however, has defined the physical characteristics (i.e. location, pin definition, electrical and mechanical specs) for what the specification calls "open system" LAN and Modem I/O connections. The open system connectors are defined in the PC Card Standard within the PCMCIA Extensions volume. The PC Card standard does not require that manufacturers implement the open system connectors, but these connectors have the advantage of a uniform interface, promoting availability of these connectors from multiple sources.

Note that the card's thickness does not permit some connectors, such as the RJ-11 and RJ-45, to be built into a type II card. As a result, pigtail extension cables are typically used for larger connectors. Other connectors allow phone jack connections without the pigtail cable, such as the Megahertz, XJack implementation.

Figure 4-3. Type II Card with External I/O Connector

PCMCIA includes a design to accommodate thicker PC Card devices (such as hard drives). The outline of type III devices is shown in figure 4-4. Most systems that have two type II socket stacked on top of one another can accept a type III card. The type III card plugs into the lower socket, but the body of the type III PC Card requires the space of both sockets.

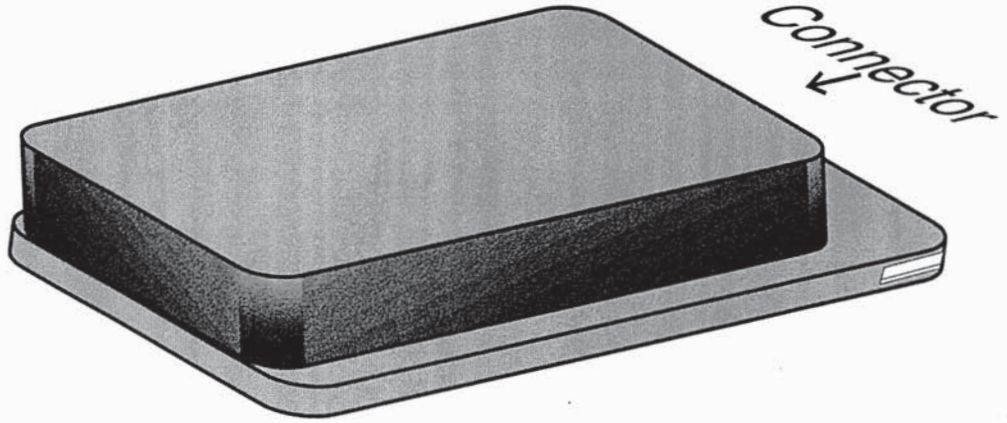


Figure 4-4. Type III Card Outline

Extended Card Types I and II

To accommodate large external connectors and to permit designers to encase their electrical and magnetic isolation devices within the shielded PC Card enclosure, PCMCIA defined extended card types I and II. Figure 4-5 shows the outline of the type I and II extended cards. Since the main body of the card is the same as the non-extended designs, these cards will fit into a typical card socket. The extension must be 10mm beyond the standard card length of 85.6mm (i.e. 95.6mm) before the height of the extension can be increased.

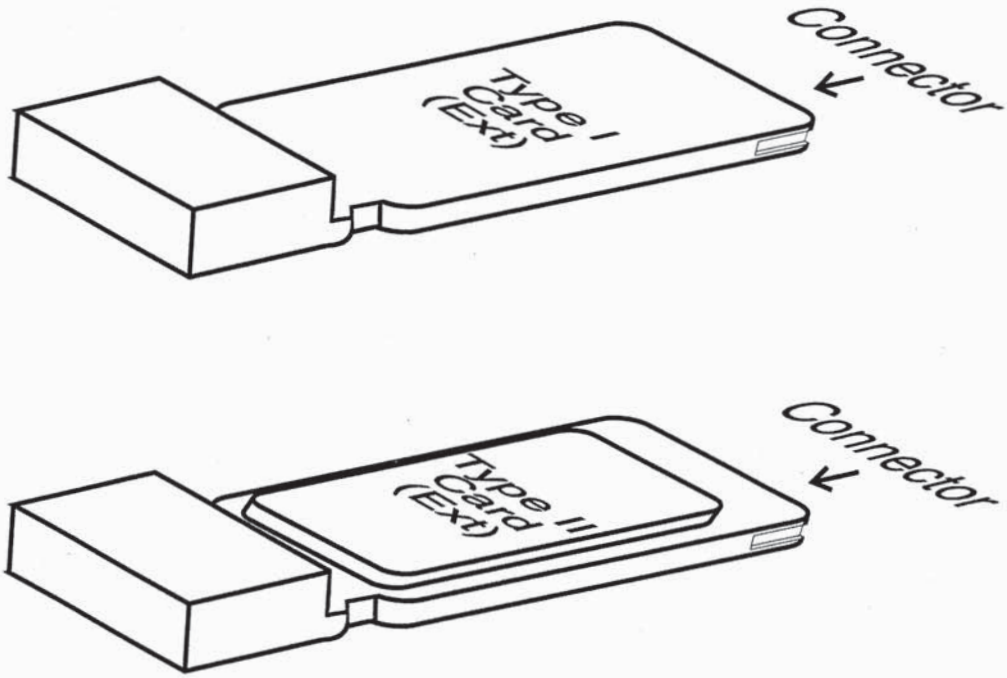


Figure 4-5. Type I and II Extended Cards

The Card and Socket Connectors

Card and Socket Keying

As shown in figure 4-6 PCMCIA cards and sockets are keyed to prevent the card from being inserted upside down. Keying is accomplished at the edges of the PC Card and the socket connector.

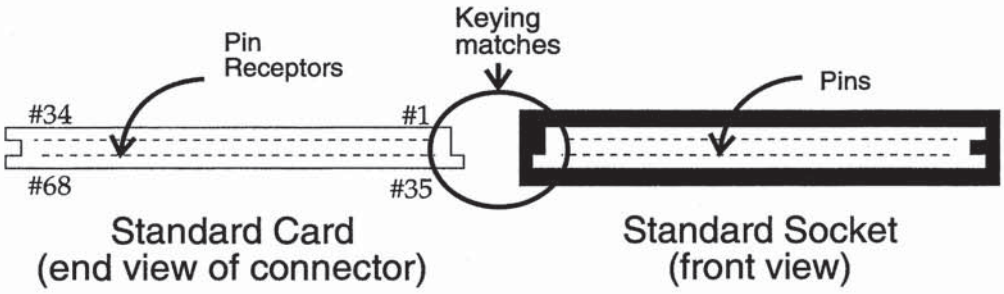


Figure 4-6. Card and Socket Keying-Standard Interface

PCMCIA release 2.0 and 2.1 cards can be designed as dual-voltage cards. These cards always power up at +5 Vcc but can switch to +3.3 Vcc for low power operation. A new generation of low-voltage cards is now possible using the new low-voltage specification. These newer low-voltage cards need not implement the dual-voltage solution described above. Instead, they can be designed for +3.3 Vcc operation only. This means that these cards will not function correctly in systems based on the 2.0 or 2.1 specification, which always apply +5 Vcc to the socket when a PC Card is first installed. Keying is employed on the newer low-voltage cards, preventing them from being inserted into standard dual-voltage sockets as shown in figure 4-7.

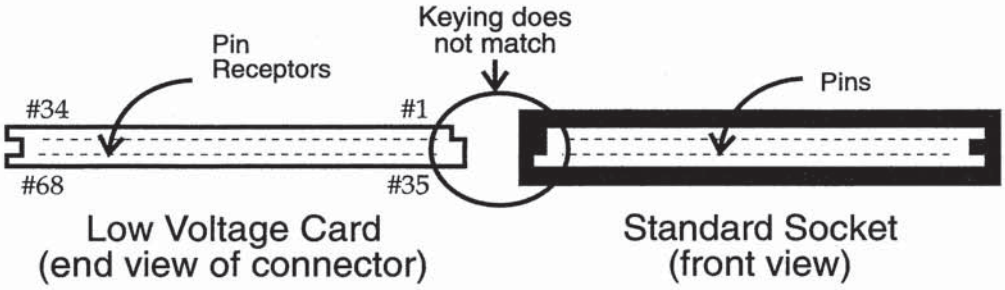


Figure 4-7. Low-Voltage Cards Cannot Be Inserted into Standard Sockets

To accommodate low voltage cards, a new low voltage socket has been designed. This socket, which might be better named the "universal socket," accepts 5 volt only cards, dual voltage cards, and 3.3vdc cards. This is possible since systems employing the low voltage socket have the ability to apply initial voltages of either +3.3 Vcc or +5.0 Vcc depending on the PC Card's requirements. Two newly defined Voltage Sense (VS) signals determine the initial voltage to be applied to the socket when a PC Card is installed. The Voltage Sense pins are set by the PC Card to indicate the initial voltage that they require.

Low voltage socket keying is shown in figure 4-8.

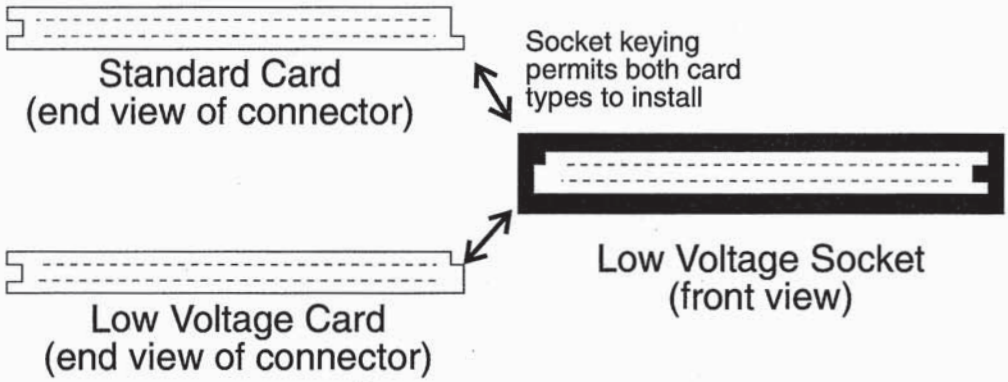


Figure 4-8. Keying Used with Low Voltage Socket

The two voltage sense pins (VS1# and VS2#) provide a means for the host system to detect the voltage level that a given PC Card must initially power up at. Refer to table 4-1. PCMCIA release 2.0 or 2.1 (2.x) compliant systems require that PC Cards always operate at 5 volts when power is first applied, and can later switch to 3.3 volts if configuration software detects that the card supports 3.3 volt operation. These PC Cards are termed dual-voltage cards. The standard PCMCIA socket designed for release 2.x systems have no voltage sense capability since all cards initially power to 5 Vcc.

PCMCIA also supports PC Cards that operate only at 3.3 volts, without initially having to power-up at 5 volts. The specification also supports cards that operate at a future non-specified low voltage, referred to as X.X volts. To accomplish 3.3 and X.X volt only operation, PCMCIA has defined a new socket that can supply the initial voltage required by new low voltage PC Cards. This socket is keyed to accept the newer low-voltage cards, 5 volt-only cards and dual-voltage cards. This is possible since the voltage sense pins determine the initial voltage that should be applied to the socket.

Table 4-1 shows the possible combinations of PC Cards that can be installed into a low-voltage socket. The first column specifies the initial voltage that the PC Card required when power is initially applied to the socket. The second column specifies the keying implemented by the card (i.e. whether the card is keyed to fit into a standard socket or a low voltage socket). Columns three and four specify the state of the voltage sense pins, and column five indicates the initial power that will be applied by the HBA. Note that X.X voltage designates a low level Vcc to be defined in the future.

Table 4-1. Interpretation of Voltage Sense Signals

Initial Power Required	Keying	VS1#	VS2#	Vcc at power up and CIS read
5 volts	standard*	1	1	5 volts applied if available, else no Vcc applied.
3.3 volts	low-voltage	0	1	3.3 volts applied if available, else no Vcc applied
3.3/5 volts	standard*			3.3 volts applied if available, else no Vcc applied.
X.X volts	low-voltage	1	0	X.X volts applied if available, else no Vcc applied.
X.X/3.3 volts	low-voltage	0	0	X.X volts applied if available, else 3.3 volts applied if available, else no Vcc applied.
X.X/3.3/5 volts	standard*			X.X volts applied if available, else 3.3 volts applied if available, else no Vcc applied.

* Standard keying refers to PC Cards keyed to fit into a 2.x compliant socket. These cards also fit into the low voltage sockets.

Pin Length

The PCMCIA Host Bus Adapter connector has three different pin lengths:

- Power pins (Ground & Vcc) — .098" (2.5 mm)
- General interface pins (address, data and control) — .084" (2.1 mm)
- Card detect pins — .059" (1.5 mm)

When a card is inserted into the host adapter's PCMCIA socket, the power pins make contact first, following by the general interface pins and then the card detect pins. Whether power will be applied to the power pins when a card is inserted depends on the system design. Many implementations apply power to the pins only after the shorter card detect pins make contact, indicating that the card is fully inserted into the HBA socket.

Environmental Characteristics

PCMCIA specifies a very thorough set of environmental tests and operating conditions under which PCMCIA cards and sockets should operate. Refer to the PCMCIA Card Standard for detailed information.

The PC Card standard specifies the following electrical and mechanical specifications for the socket connector and for PC Cards. The following tables list the primary specifications. For a complete list of mechanical and electrical specification and more information regarding the testing criteria and methods see the PC Card Standard's Physical Specification.

Connector Environmental Standards

Table 4-2 is a partial list of specifications defined by the PC Card Standard for connector reliability. All tests and measurements are specified to be made at the following ranges unless otherwise specified:

- temperature — 15ºC to 35ºC
- air pressure — 86 to 106 kPa
- relative humidity — 25% to 85%

Table 4-2. Selected Connector Reliability Specifications

Parameter	Standard/Specification	Testing Criteria
Operating Environment	Operating temp -20º to +60ºC Relative humidity, 95%	
Storage Environment	Storage temp -40º to +70ºC Relative humidity, 95%	
Number of insertions and ejections	10,000	Office Environment (see EIA-364-B Class 1.1)

Table 4-2. Selected Connector Reliability Specifications

Parameter	Standard/Specification	Testing Criteria
Number of insertions and ejections	5,000	Harsh Environment (see EIA-364-B Class 1.1)
Total insertion force	39.2 N maximum	Insert at 25mm/minute
Total pulling force	6.67N min. / 39.2 N max.	Extract at 25mm/minute
Single pin pulling force	Pin shall remain in insulator when .098 N of pulling force is applied.	Pull at 25mm/minute
Single pin holding force	Pin shall remain in insulator when 9.8 N of pushing force is applied.	Push at 25mm/minute
Single socket holding force	Socket shall not be dislodged or damaged when 4.9 N force is applied.	Push socket on axis at 25mm/minute.
Vibration and Frequency	No mechanical damage & no current interruption > 100ns.	MIL-STD-202F Test @ 147m/s^2,10 - 2000Hz
Shock	No mechanical damage & no current interruption > 100ns.	MIL-STD-202F Test @ 409m/s^2 acceleration
Contact Resistance (low level)	40mΩ max (initial value) 20mΩ max change (after test)	MILSTD-1344A Open voltage 20mV, Test current 1mA
Withstanding voltage	No shorting or damage when 500Vrms AC is applied for 1 minute, current leakage of 1mA max.	MIL-STD-202F
Insulation resistance	1000MΩ min (initial value) 100MΩ min (after test)	MIL-STD-202F Apply 500vdc
Current capacity	0.5 A per contact	Based on 30ºC rise above ambient temperature
Insulation Material	UL 94 V-0	UL Standard 94
Ground Return Inductance	18.0 nH max @ 1MHz	ANSI/EIA-364-69

PC Card Environmental Standards

Table 4-3 lists selected environmental specifications for the PC Card. All tests and measurements are specified to be made at the following ranges unless otherwise indicated:

- temperature — 15ºC to 35ºC
- air pressure — 86 to 106 kPa
- relative humidity — 25% to 85%

Table 4-3. Selected PC Card Environmental Specifications

Parameter	Standard/Specification	Testing Criteria
Operating Environment	Operating temp 0º to +55ºC Max. relative humidity, 95%	96 hours minimum
Storage Environment	Storage temp -20º to +65ºC Max relative humidity, 95%	96 hours minimum
Thermal Shock	Data to be retained by all non-volatile memory after test	MIL-STD-202F Method 107G Test 1 @ -20ºC for 30 min. Test 2 @ 25ºC for <05 min. Test 3 @ 65ºC for 30 min. Test 4 @ 25ºC for <05 min. Repeat all 4 test for 100 cycles
Moisture Resistance	PC Card must function as specified after test and must retain data stored in non-volatile memory prior to test.	MIL-STD-202F Method 106E Test 1 @ 20 to 65ºC for 2.5 hrs. Test 2 @ 65ºC for 3 hrs. Test 3 @ 65 to 25ºC for 2.5 hrs. Test 4 @ 25 to-10ºC for 2.5 hrs. Test 5 @ -10ºC for 3 hrs Test 6 @ -10 to 25ºC for 2.5 hrs Repeat all 6 tests for 10 cycles
Electrostatic Discharge	PC Card must retain data stored in non-volatile memory prior to test.	ISO 7816-1 See Mechanical Specification
X-ray Exposure	PC Card must function as specified after test and must retain data stored in non-volatile memory prior to test.	ISO 7816-1 Wavelength 254 nm Intensity 15000 $\mu W/cm^2$ Exposure time 20 min
EMI	PC Card must function as specified after test and must retain data stored in non-volatile memory prior to test.	ISO 7816-1 Uniform magnetic field of 1000 Oersted Exposure time=10 seconds 100 Oersted Exposure time-Exposure time=10 seconds for rotating media cards.

Table 4-3. Selected PC Card Environmental Specifications (continued)

Parameter	Standard/Specification	Testing Criteria
Card Inverse Insertion	No electrical contact between card and connector except Vcc and ground pins.	See Mechanical Specification
Vibration and High Frequency	PC Card must function as specified after test and must retain data stored in non-volatile memory prior to test.	MIL-STD-202F Method 204D Test B: 147 m/s^2 peak, 10 to 2,000 Hz, 12 cycles per axis, 36 cycles for 3 axes (12hrs). Battery installed, no Vcc.
Shock	PC Card must function as specified after test and must retain data stored in non-volatile memory prior to test.	MIL-STD-202F Method 213B Test Condition A: 490 m/s^2 Standard holding time 11 ms Semi-sine wave
Bend Test	PC Card must function as specified after test and must retain data stored in non-volatile memory prior to test. Dimensions must conform to use requirements after test.	See Mechanical Specification.
Drop Test	PC Card must function as specified after test and must retain data stored in non-volatile memory prior to test. Dimensions must conform to use requirements after test.	See Mechanical Specification.
PC Card Warpage	PC Card must function as specified after test and must retain data stored in non-volatile memory prior to test. Dimensions must conform to use requirements after test.	See Mechanical Specification.

Chapter 5

The Previous Chapter

The previous chapter focused on the various physical packages defined by PCMCIA for PC Cards and the related environmental specifications. The chapter also described the standard and low-voltage socket connector types.

This Chapter

This chapter details the memory-only electrical interface between the PC Card and socket. Each pin is defined and its relationship to the PC Card and the HBA is discussed. The memory-only interface is the interface initially seen by 16-bit PC Cards when they are first inserted into a socket. This permits the memory-mapped CIS to be accessed to determine the PC Card type and interface requirements. If the card is designed for an interface type other than memory-only, then the HBA and PC Card are configured to communicate via one of the other interfaces defined by the PC Card Standard (discussed in the following chapters). Also discusses the timing of socket accesses to PC Cards of differing speeds, including transfers with attribute memory and common memory.

The Next Chapter

The next chapter discusses the electrical interface called the memory or I/O PC Card and Socket interface. This interface is used by standard I/O-based PC Cards.

Overview

The original PCMCIA standard (release 1.0) defined the socket interface for memory cards only. Later releases added additional socket interfaces. The PC Card Standard defines the following socket interfaces:

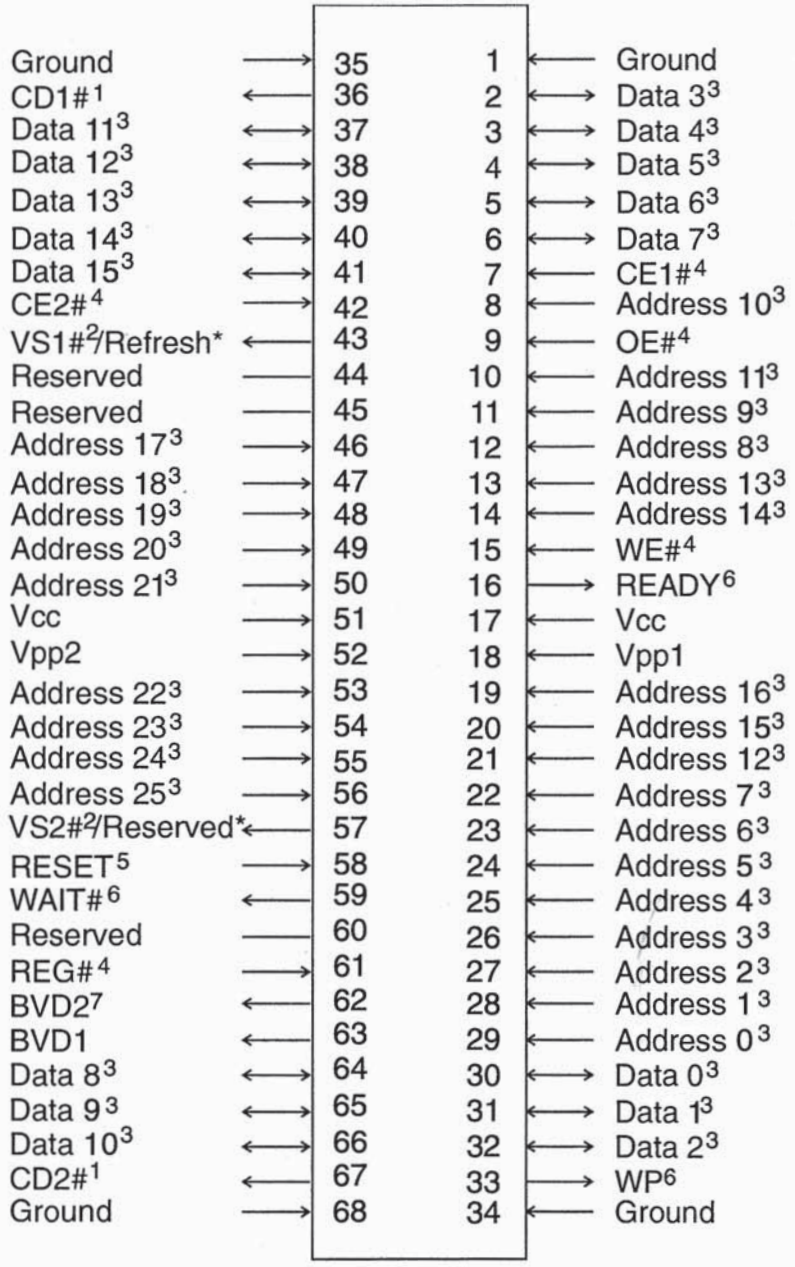
- Memory only
- Memory or I/O
- ATA (AT attachment for IDE drives)
- DMA (Direct Memory Access)
- AIMS (Auto-Indexing Mass Storage)
- CardBus (32-bit PC Card interface)

Each of these interfaces employs the PCMCIA 68 pins connector. However pin definition changes with each interface type. Most HBAs can be configured to support a variety of socket interfaces. This chapter discusses the memory-only interface. Subsequent chapters discuss the other socket interfaces included in the bulleted list above.

This chapter discusses the PCMCIA memory interface, detailing the electrical interface used PC Cards that employ only a memory function. However, all 16-bit PC Cards must initially operate as memory-only devices. This is necessary since the HBA has no way of knowing whether a PC Card installed required a specific interface other than memory. Therefore, HBAs always present a memory-only interface when a PC Card is initially installed, and all 16-bit PC Cards must operate as memory-only devices when they are inserted into a PC Card socket. A card's client driver is then responsible for reading from the PC Card's CIS (accessible via the memory-only interface) to determine the card type and the socket interface that it requires. The HBA and PC Card can then be dynamically reconfigured under software control to use the specified socket interface. Note however that not all HBAs are designed to support all socket interface types.

The Memory Interface

Two types of memory address space exist within a PC Card: common memory and attribute memory. *Common memory* is the working address space used to map the memory arrays that typically store data and executable files. *Attribute memory* is used for configuration information. The attribute memory address space contains the CIS (Card Information Structure) and configuration registers. Figure 5-1 illustrates the signals that comprise the memory interface which provide access to both common and attribute memory. The signals defined for the memory interface are grouped functionally and described in the following sections.



1. Pulled-up to Vcc by HBA (R≥10KΩ).
2. Pulled-up to Vcc by HBA (R=10KΩ-100KΩ).
3. Pulled-down by PC Card (R≥100KΩ).
4. Pulled-up to Vcc by PC Card (R≥10KΩ).
5. Pulled-up to Vcc by PC Card (R≥100KΩ).
6. Pulled-up to Vcc by HBA (R≥10KΩ).
7. Pulled-up to Vcc by HBA.

Figure 5-1. PCMCIA Memory Socket Interface to Host Bus Adapter

Card Power

PC Card memory sockets provide the standard logic supply voltage (Vcc) and programming voltages (Vpp) typically used by EEPROM and Flash devices. Table 5-1 lists the power pins defined by the memory socket.

Table 5-1. Card Voltage Pins

Signal	Function
Vcc & Ground	Card Voltage. Two Vcc pins and four ground pins are used with each socket. Vcc will always be 5 volts when the card is first accessed if the socket is based on the 2.x standard. In these system, if the PC Card supports dual operating voltages (5 volts and 3.3 volts) the operating voltage will be reduced to 3.3vdc during card initialization. The PC Card standard specifies that each Vcc pin supply a maximum of 500mA (1A total per socket). The actual amount of current that can be supplied by the HBA is design specific. Systems that support low voltage sockets determine the initial Vcc that is applied to the socket by sampling VS1# and VS2#.
Vpp1 & Vpp2	Programming Voltage. These pins are used for special programming voltages that may be required by programmable memory devices. The card's CIS must specify the required programming voltage. Each Vpp pin must be able to supply 30mA of current.

Two types of sockets are defined by PCMCIA: the standard 2.x compliant socket and a new low voltage (universal) socket. The initial Vcc voltage that a socket can apply to a PC Card depends on the socket type and the HBA design. Figure 5-2 illustrates the voltage switching that PCMCIA HBAs implement if they support the VS1# and VS2# signals.

Release 2.x Socket

The card power pins are listed in table 5-1. For systems that are 2.x compliant, 5 volts is always applied to the Vcc pin when a PC Card is first installed into a socket. Vcc can later be switched to 3.3 volts if the PC Card has dual voltage capability. Support for 3.3vdc operations is detected by a PC Card's client driver when the CIS is interrogated. It should be apparent that any card inserted into a 2.x compliant socket must be able to operate at 5 volts.

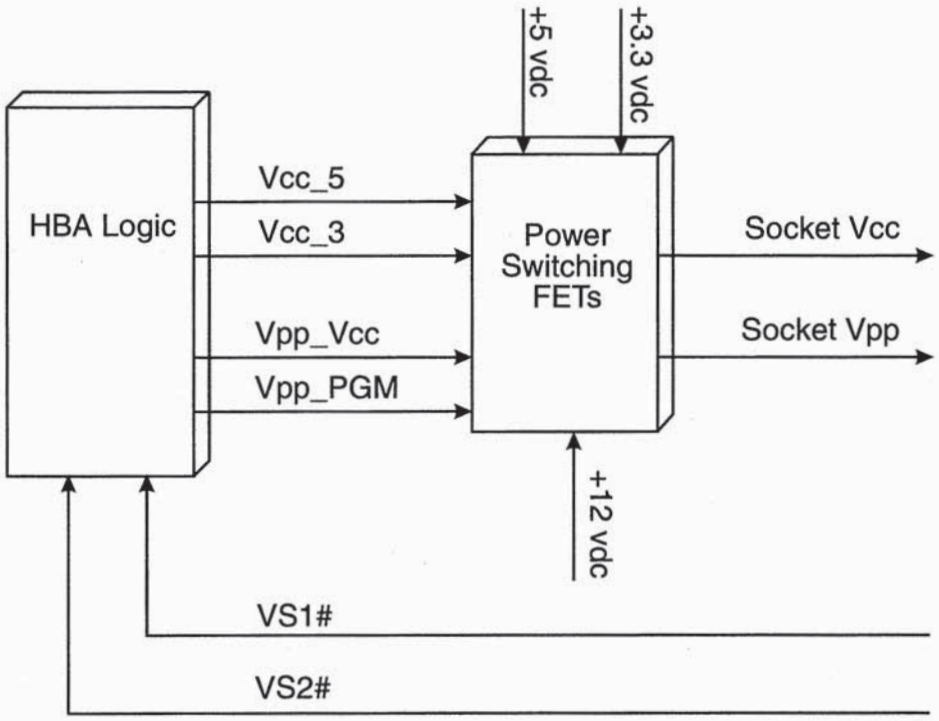


Figure 5-2. Voltage Switching Performed by HBA

Low-Voltage Socket

In systems that implement low-voltage sockets, the initial Vcc applied to the PC Card can be either X.X, 3.3 or 5 volts. The initial voltage is determined by the state of voltage sense pins that have been defined for the low voltage socket. Refer to the chapter entitled "PC Card and Socket Physical Design" for details on the low voltage socket.

Voltage Sense Pins (not used in 2.x systems)

Two Voltage Sense pins (VS1# and VS2#) provide a means for the host system to detect the voltage level that a given PC Card must initially power up at. Refer to table 5-2. PCMCIA release 2.0 or 2.1 (2.x) compliant systems always apply 5 volts when power is first applied, and can later switch to 3.3 volts if configuration software detects that the card supports 3.3 volt operation (via the CIS). These PC Cards are termed dual-voltage cards. The standard

PCMCIA socket designed for release 2.x systems has no voltage sense capability since all cards initially power to 5 Vcc.

Table 5-2. Definition of Voltage Sense Pins

VS1#/Refresh	**Voltage Sense 1 or Refresh**. The values of VS1# and VS2# determine the initial Vcc voltage that is applied to the socket when a PC Card is installed. Both VS pins are pulled to 5 volts by the host system. When a low voltage card is installed, it pulls one or both of the VS pins low to indicate the initial Vcc it requires. When a standard 2.x compliant PC Card is inserted into a low-voltage socket, both VS1# and VS2# will remain asserted (because 2.x cards do not implement the VS pins). A 2.x compliant socket defines the VS1# pin as the refresh signal, originally intended to be used with pseudo-static RAM. Its functionality has not been defined and is not used.
VS2#/Reserved	**Voltage Sense 2 or Reserved**. This pin is defined as VS2# for low-voltage sockets and cards and is pulled to 5 volts by the host system. See the description for VS1# above. This pin is reserved in a 2.x compliant socket.

PCMCIA also supports PC Cards that operate only at 3.3 volts (i.e. they do not initially power-up at 5 volts). The PC Card standard also supports cards that operate at a future non-specified low voltage, referred to as X.X volts. To accomplish 3.3 and X.X volt only operation, PCMCIA defines a PC Card socket that can supply the initial voltage required by 3.3vdc only PC Cards. This socket is keyed to accept the newer low-voltage cards, as well as, 5 volt-only cards and dual-voltage cards. This is possible because the HBA samples the voltage sense pins to determine the initial voltage required by the PC Card.

Keying on the low-voltage cards prevent them from being inserted into standard 5 volt sockets, thus avoiding the circuit damage that could result if a 3.3vdc card was installed in a 2.x compliant system, which always supplies 5vdc initially to the socket. In summary two types of sockets are specified by PCMCIA.

- Standard Socket — keyed to accept cards that can power up at 5 volts. These sockets accept 5 volt-only cards and cards that can be switched to 3.3 volt operation.

- Low Voltage Sockets — keyed to accept either 3.3 volt, X.X volt or 5 volt cards. Cards designed for 3.3 volt-only operation are keyed to fit only into

this socket, thus preventing possible damage. Note that systems implementing the low-power socket monitor the voltage sense lines; therefore, they can accept PC Cards operating at any of the specified voltages.

Table 5-3 shows the possible combinations of PC Cards that can be installed into a low voltage socket and the resulting Vcc that will be initially applied to the card based on the state of VS1# and VS2#. The X.X voltage designates a Vcc level to be defined in the future.

Table 5-3. Interpretation of Voltage Sense Signals by a Low Voltage Socket

Initial Power Required	Keying	VS1#	VS2#	Socket Vcc at power up
5 volts	standard[1]	1	1	5 volts applied if available, else no Vcc applied.
3.3 volts	low-voltage	0	1	3.3 volts applied if available, else no Vcc applied
3.3/5 volts	standard[1]	PD[2]	1	3.3 volts applied if available, else no Vcc applied.
X.X volts	low-voltage	1	0	X.X volts applied if available, else no Vcc applied.
X.X/3.3 volts	low-voltage	0	0	X.X volts applied if available, else 3.3 volts applied if available, else no Vcc applied.
X.X/3.3/5 volts	standard[1]	PD[2]	0	X.X volts applied if available, else 3.3 volts applied if available, else no Vcc applied.

1. Standard keying refers to PC Cards keyed to fit into a 2.x compliant socket. These cards also fit into the low voltage sockets.
2. PD indicates that the card connects VS1# to ground via a 1KΩ ±10% pull-down resistor.

The Power-Up Sequence

When the HBA recognizes that a PC Card has been inserted into a socket (has detected CD1# and CD2# asserted) it must ensure that the card is powered and the interface signals are enabled in the correct sequence. The PC Card standard specifies the timing and power sequences required. Figure 5-3 illustrates the power-up sequence and the primary timing parameters. Note that RESET is asserted no sooner than 1ms after Vcc has stabilized. Once RESET is deasserted a PC Card has 20ms to perform its initialization before the system

begins accessing the card. If the card needs longer than 20ms before being accessed, it must deassert READY until it is ready to be accessed.

Note that during intial power-up the PC Card is not permitted to draw more than 70mA at 3.3vdc or 100mA at 5.0vdc. This limit must be maintained even if the PC Card requires additional current during normal operation.

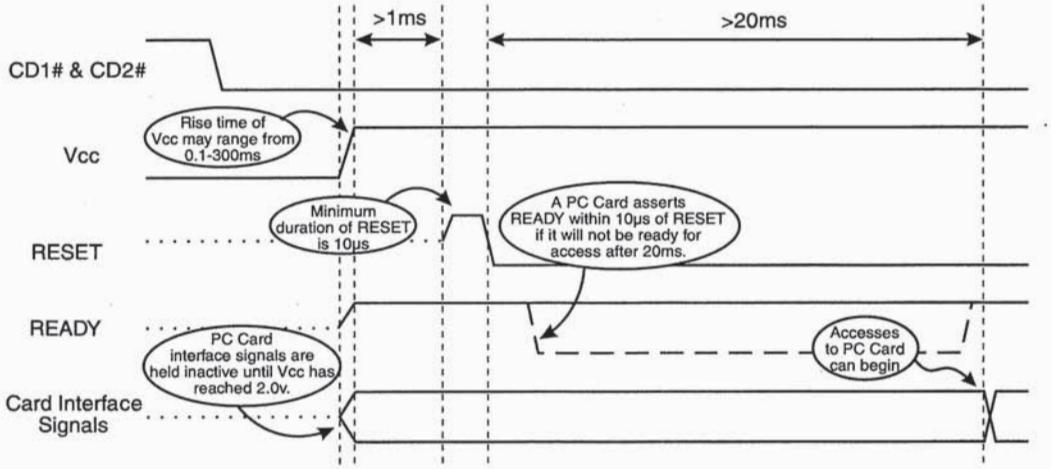


Figure 5-3. The Socket Power-up Sequence

Vpp1 and Vpp2

Vpp1 and Vpp1 (Programming or Peripheral voltages) provide the programming voltage needed by devices such as EEPROM (Electrically-Erasable Programmable ROM) or Flash ROM during write operations. The PC Card standard permits separate voltages to be applied via the Vpp1 and Vpp2 pins; however, this ability may not be supported by given HBA implementations.

When the socket is first powered, Vpp1 and Vpp2 are connected to Vcc. After the CIS is read, an alternative Vpp can be applied to the socket under software control. Twelve volts is recommended as a an alternative Vpp voltage that can be supplied by the HBAs; however, other voltages may also be applied depending on the card's requirements specified in the CIS and the capabilities of the HBA.

Address Signals

Refer to table 5-4. The card address signals consist of the address bus proper (A25:A0) and the card enable signals (CE1# and CE2#). When a transaction to a PC Card begins, the address lines along with CE1# and CE2# are asserted. When both CE1# and CE2# are deasserted, the card is in standby mode (waiting to be accessed). PCMCIA supports 16-bit PC Cards in both 8-bit hosts (e.g. Intel 8088-based systems) and 16-bit host (e.g. ISA expansion buses). The card enable signals addressing modes compatible with both 8- and 16-bit host systems.

Table 5-4. PCMCIA Address Lines—Memory Interface

Signal	Function
A25:A0	**Address Lines 0-25**. Twenty-six address lines permit a total address space of 64 MB for PCMCIA cards. This 64 MB address space is valid for both common memory and attribute memory.
CE1#	**Card Enable 1**. The CE1# signal is an active low signal that specifies access to address locations which are transferred over the lower data path (D7:D0). During accesses by 16-bit hosts, CE1# specifies even location access only. During accesses by 8-bit hosts CE1# specifies access to either an even or odd location, with A0 determining whether an even or odd location is being addressed.
CE2#	**Card Enable 2**. The CE2# signal is an active low signal that specifies access to an odd address location during access by 16-bit hosts. When CE2# is asserted, valid data is always transferred over the upper data path (D15:D8). During access by an 8-bit host, CE2# is always deasserted. When CE1# is deasserted and CE2# is asserted, an odd byte is being accessed by a 16-bit host, and when both CE1# and CE2# are asserted, two bytes are accessed from the card by a 16-bit host.

Table 5-5 identifies the combination of A0, CE1# and CE2# used to access even and odd address locations from PC Cards and shows which data path is used. This addressing scheme permits access to PC Cards by both 8-bit and 16-bit host systems. For example, note that there are two ways to access an odd byte from a PC Card, depending on whether the host system is an 8-bit system (uses only D7:D0) or a 16-bit system (uses D15:D8 and D7:D0).

Table 5-5. Addressing Even and Odd Bytes

Addressing Mode	CE1#	CE2#	A0*	D15:D8	D7:D0
No access (standby mode)	1	1	X	High-Z	High-Z
8/16-bit Mode (even byte)	0	1	0	High-Z	Even Byte
8-bit Mode (odd byte)	0	1	1	High-Z	Odd Byte
16-bit Mode (odd byte only)	1	0	X	Odd Byte	High-Z
16-bit Mode (even & odd byte)	0	0	X	Odd Byte	Even Byte

* "X" indicates a don't care condition.

When a PC Card is accessed in 16-bit mode (by 16-bit hosts), CE1# indicates access to an even address location, while CE2# indicates access to an odd address location (Refer to figure 5-4). Even locations are transferred over data path D7:D0, and odd locations over path D15:D8..

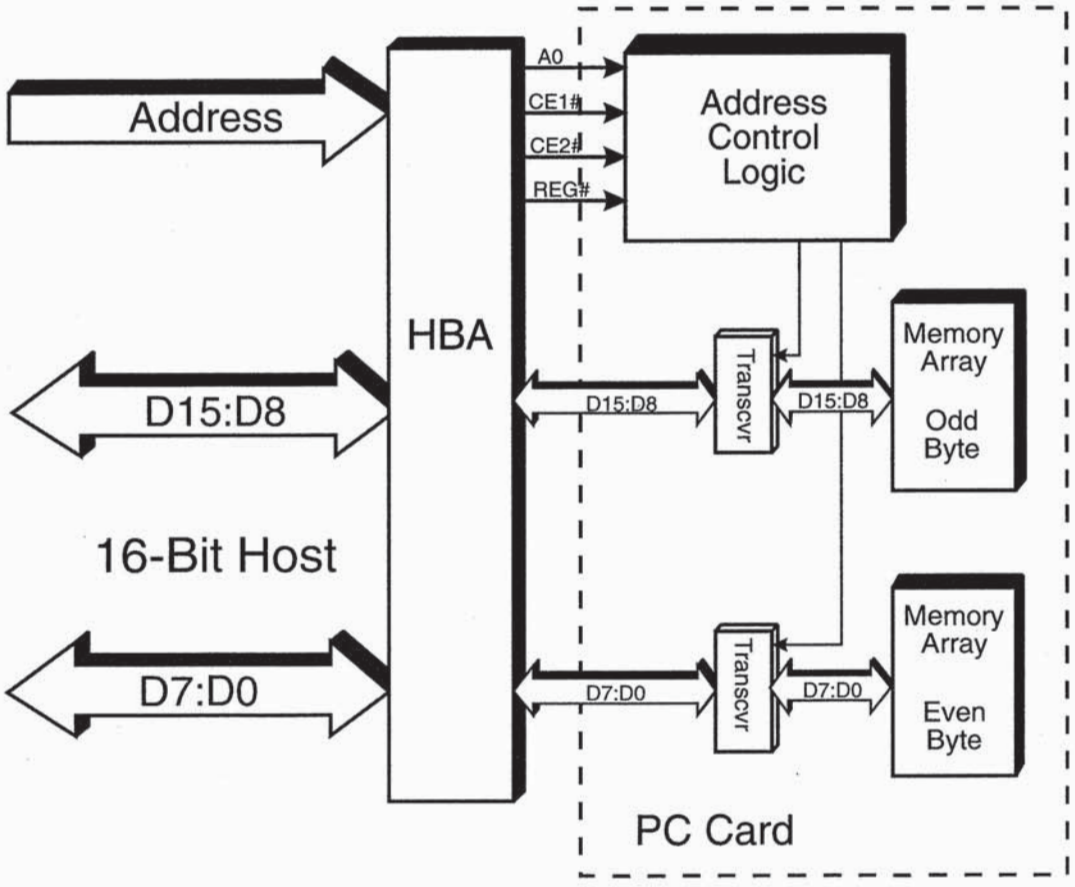


Figure 5-4. Addressing Mode Used by Memory Card with 16-Bit Host

When a PC Card is accessed in 8-bit mode (by an 8-bit host), both even and odd locations must be transferred to or from the PC Card over path D7:D0 (Refer to figure 5-5). In this case, CE2#, CE1# and A0 control data bus steering within the PC Card, ensuring that both even and odd locations are transferred over data path D7:D0.

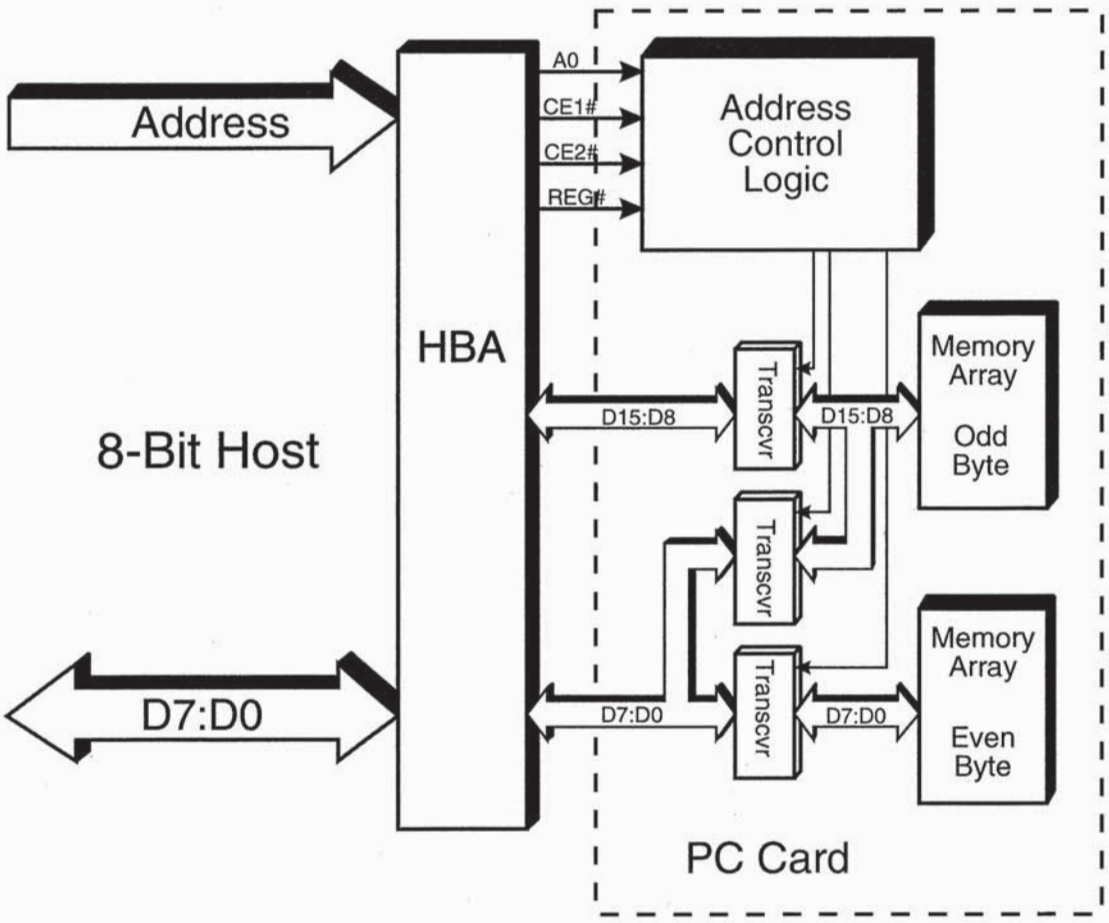


Figure 5-5. Addressing Mode Used by Memory Card with 8-Bit Host

Note that attribute memory devices (i.e. the CIS) only contain valid data at even address locations (Refer to figure 5-6). This simplifies the design of attribute memory when accessed by 8-bit hosts (expansion buses). That is, no data bus steering logic need be implemented, since even address locations are only accessed over D7:D0.

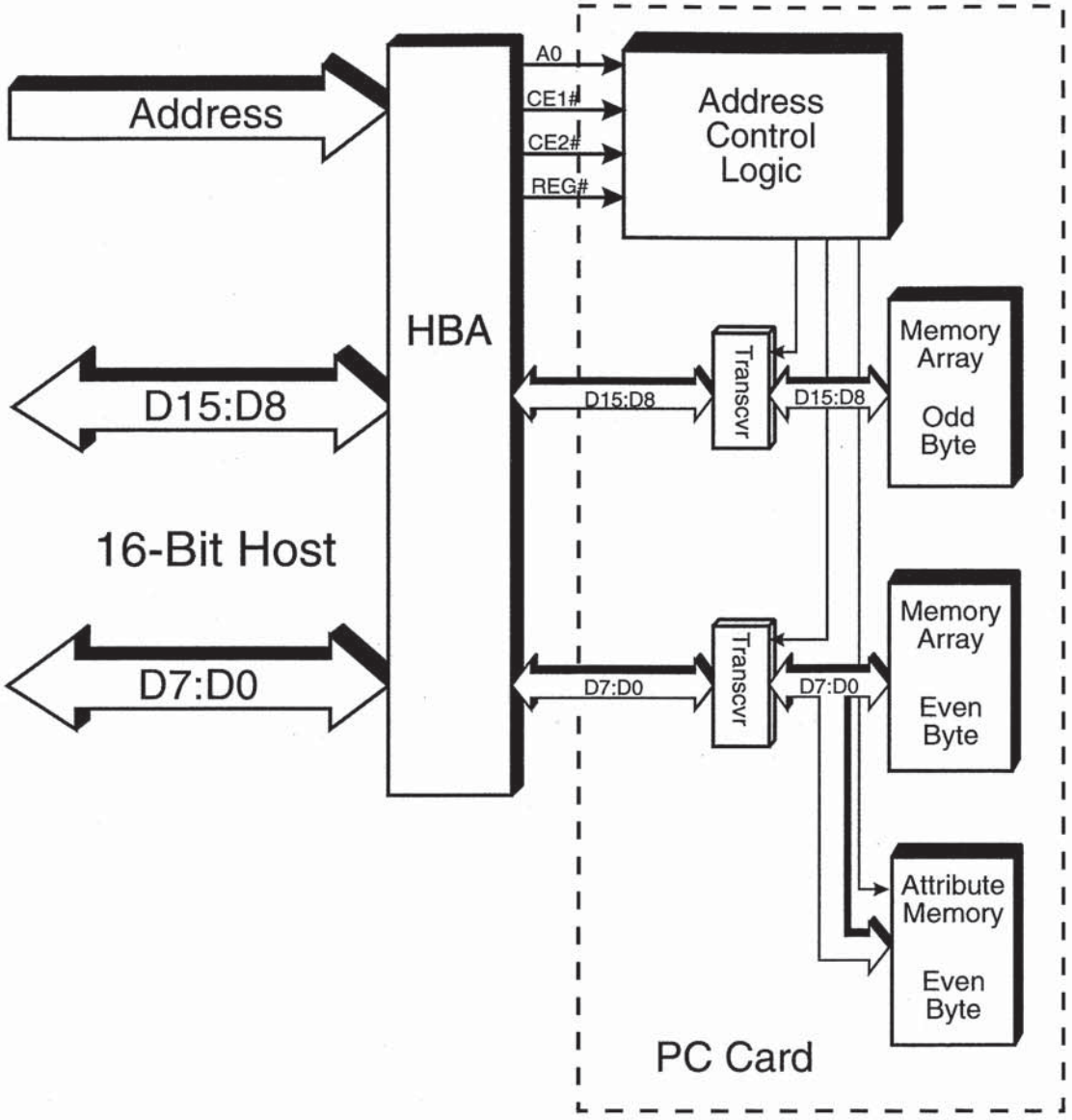


Figure 5-6. Only Even Locations Are Accessed from Attribute Memory over the Lower Data Path

Data Lines

The data lines consist of two 8-bit paths; the lower path, data lines D7:D0 and the upper path, data lines D15: D8 (See Table 5-6.) The data path(s) carrying valid data during a data transfer are defined by the host data bus size and whether an even byte, odd byte or word is being transferred. Note that PCMCIA supports PC memory cards containing only 16-bit devices.

Table 5-6. Data Bus

Signal	Function
D7:D0	**Data lines 7:0** (Lower data path). Transfers data to and from even locations when the host system has a 16-bit data path; transfers data to and from both even and odd locations when host system has a single 8-bit data path.
D15:D8	**Data lines 15:8** (Upper data path). Transfers data to and from odd locations when the host system has a 16-bit data path; not used when the host system has a single 8-bit data path.

PC Memory Card Transaction Definition

Four types of transactions can be performed when accessing a PC Card memory locations:

- Common Memory Read
- Common Memory Write
- Attribute Memory Read
- Attribute Memory Write

The Output Enable signal (OE#), Write Enable or Program signal (WE#) and Register (REG#) signals define the transaction type. Table 5-7 defines each type of PC Card memory command. Table 5-8 shows the possible command signal combinations for common and attribute memory.

Table 5-7. PC Card Command Lines for Memory Interface

Signal	Function
OE#	**Output Enable**. This signal is active low and is asserted during memory read transfers from PCMCIA cards.
WE#	**Write Enable/Program**. This signal is active low and is asserted during memory write transfers to PCMCIA cards. When the PC memory device requires a special programming voltage to perform a write operation, this signal is defined as the "program" command signal.
REG#	**Register Select**. This signal when asserted specifies access to attribute memory.

Table 5-8. PCMCIA Memory Transaction Types

Transaction Type	OE#	WE#	REG#
Common Memory Read	0	1	1
Common Memory Write	1	0	1
Attribute Memory Read	0	1	0
Attribute Memory Write	1	0	0

PC Memory Card Status Signals

PCMCIA supports status reporting for a number of card conditions or events including:

- Card Detection (CD1# and CD2#)
- Ready/busy (READY)
- Write-Protect (WP)
- Low Battery Detection (BVD1 and BVD2)

As described in table 5-9, the memory interface has dedicated signals used for reporting status back to the HBA. Additional information on each of the status conditions is provided in the following section.

Table 5-9. Card and Socket Status Signals

Signal	Function
CD1#	**Card Detect 1.** This signal is asserted while a PC Card is installed in a socket, indicating that one end of the card is making electrical contact within the socket. This pin should be pulled up to 5 volts by the host system.
CD2#	**Card Detect 2.** This signal is asserted while a PC Card is installed in a socket, indicating that the opposite end of the card is making electrical contact within the socket. This pin should be pulled to 5 volts by the host system.
READY	**READY.** Used by the card to inform the system that the card is ready to be accessed. When READY is deasserted the card is processing a command or performing initialization and no access should be attempted to the card until READY is asserted again.
WP	**Write-Protect.** This signal reports the status of the card's write protect switch (if present). If no write protect switch is used the WP pin should be pulled up to Vcc (read only) or pulled low (writeable) based on its read/write capabilities.
BVD1	**Battery Voltage Detect 1.** In conjunction with BVD2, indicates the status of the PC Card's battery, if present.
BVD2	**Battery Voltage Detect 2.** In conjunction with BVD1, indicates the status of the PC Card's battery. JEDEC compliant cards use only BVD1 to report battery status. The host system should pull BVD2 to Vcc on the card to maintain compatibility with JEDEC cards. Otherwise, a JEDEC compliant system will detect a low battery condition which none exists.

* Note that systems based on Release 2.0 or 2.1 of the PC Card Standard do not include the voltage sense capability. These systems always supply 5vdc to a PC Card when it is initially inserted into a socket.

Card Detection

The card detection signals, CD1# and CD2# provide a method of notifying the host system when a PC Card has either been inserted or removed. The host system generates an interrupt when a change in the card detection signals occurs. Host software must take the appropriate action to either configure the PC Card that has been inserted, or to deallocate system resources that were previously assigned to a PC Card that has been removed.

Table 5-10 defines how the card detect signals should be interpreted.

Table 5-10. Interpretation of the Card Detect Signals

CD2#	CD1#	CD Status
1	1	No card inserted
1	0	Card partially inserted
0	1	Card partially inserted
0	0	Card fully inserted

Ready Status

READY is a status signal that indicates when the card is ready to be accessed. When this signal is asserted low, the card is busy and should not be accessed. When the signal is asserted high, the card is ready for access.

The busy condition typically occurs during system initialization if the PC Card requires more than 20 ms to initialize after the reset signal is deasserted. When reset is deasserted, software can normally access a PC Card after a 20ms delay; however, if the initialization is not yet completed by the PC Card, then the busy signal is asserted to prevent premature access to the card. Upon completion of initialization, the READY signal transitions to the ready state.

Busy should also be indicated during normal operation if the PC Card is processing a command or performing some lengthy operation, during which the card cannot be accessed.

Changes in the READY status should generate a system interrupt to notify software that a given PC Card is busy and should not be accessed, or that the PC Card was previously busy and is now ready for access. Software should hook the status interrupt to gain notification of READY status changes. If the interrupt is not hooked by software, then it should periodically poll the READY status.

Write-Protect Status

Some memory cards have write-protect capability. Normally a manual switch is incorporated on the end of the card that selects whether the memory is to be write-protected or not, as reflected by the card's WP signal. When a change in the write-protect status occurs the WP signal is asserted. The HBA senses WP asserted and may (depending on the software defined interrupt mask bit) initiate a status change interrupt to notify software of the change.

The PC Card designer can define which address ranges it wishes to allow the user to write protect and which ranges are always writable. This definition is included in the Card's CIS. Configuration software reads the CIS to determine which ranges of the PC Card's address space can be write protected and programs the HBA so that it will block writes to the specified address ranges when the WP pin is asserted. The state of the WP pin has no effect on writes to address ranges that the CIS defines as always writable.

Low Battery Detection

Two signals (BVD1 and BVD2) report status of PC Card batteries. Batteries are typically implemented on PC Cards that contain volatile memory devices, such as SRAM cards. The PC Card should pull-up both BVD1 and BVD2 to Vcc, indicating a fully operational battery. To indicate a low battery warning or a dead battery the PC Card deasserts these signals as shown in table 5-11.

The JEIDA memory card specification supports low battery detection using only the BVD1 signal. When a JEIDA compliant card is inserted in a socket, it will not pull BVD2 to Vcc, allowing BVD2 to float. If the HBA interprets the BVD2 signal as a logic low, then a PCMCIA compliant system would report a low battery warning when in fact no such condition exists. To alleviate this potential problem, systems designed to support JEIDA memory cards should pull-up BVD2 to Vcc in the HBA.

Table 5-11. Interpretation of Battery Voltage Detection Signals

BVD2	BVD1	Battery Status
1	1	Battery in good condition
1	0	Battery cannot maintain data integrity
0	1	Battery replacement warning — data integrity maintained
0	0	Battery cannot maintain data integrity

Bus Cycle Control

Cards use the WAIT# signal to extend normal access timing to the card. That is, a data transfer between the host bus and a PC Card can be stretched out by the PC Card when it asserts the WAIT# signal.

Normally, when accessing a PC Card, the speed of the transaction is defined by the device speed specified within the card's CIS. Configuration software reads the PC Card's speed and configures the HBA to run transactions to the PC Card based on the programmed value. When the host system accesses the PC Card, the HBA initiates a card transaction and accesses the PC Card at its specified speed.

A PC Card that asserts the WAIT# signal can extend the transfer, if it is unable to complete the transfer within the normal timing. The maximum value that wait can extend the cycle timing can be specified within the CIS in the configuration entry tuple (tuple code 1Bh). The maximum WAIT# duration allowed by the specification is 12µs.

Card Reset

The RESET signal forces the PC Card to reset internal devices and clear its configuration registers. (Note that many memory-only PC Cards do not implement configuration registers.) Reset is typically derived from the system's master reset signal, but can also be asserted under software control via an HBA register or the RESET bit in card's configuration register, if implemented.

The host system holds RESET in a high-impedance state during PC Card power-up. This occurs during system power-up if the card is installed when the host system is turned on, or when the card is installed with power already applied to the system. RESET must remain in high impedance for at least 1 ms after Vcc becomes valid. When RESET is asserted it should remain active for 10µs as specified by the PCMCIA standard.

PC Card Memory Transfers

The PC Card standard specifies standard cycle timing for accessing attribute memory and common memory devices residing within PC cards. These cycle times indicate that a PC card can be accessed at intervals equal to the speed rating specified by the card's CIS. Cycle time includes the setup, command and recovery time required to access a PC Card.

This section focuses on the relationships and functions of the signals used to control access to PC cards, and does not attempt to define all timing parameters and minimum and maximum values. This is the province of the PCMCIA specification.

Table 5-12 lists the specified cycle times for attribute and common memory. Note that attribute memory timing is 300ns, while cycle time for common memory devices range from 100ns to 250ns. A special 600ns cycle time is included for 3.3vdc memory cards requiring slower cycle time.

Table 5-12. Standard Cycle Times for PC Card Memory Devices

Memory Type / Speed	600ns	300ns	250ns	200ns	150ns	100ns
Attribute Memory		X				
Common Memory (5V)			X	X	X	X
Common Memory (3.3V)	X		X	X	X	X

Attribute Memory Read Transfers

The first access made to a PC card is a memory read from the Card Information Structure (CIS), which is mapped into *attribute* memory address space. Address zero is the first location read from, followed by the next even location. Only even locations are accessible within attribute memory space, making it easier to accommodate 8-bit host accesses. This means that only the lower data path (D7:D0) contains valid data when reading from and writing to attribute memory locations. Table 5-13 highlights the only supported addressing mode for attribute memory accesses.

Table 5-13. Addressing Mode Supported by Attribute Transfers

Addressing Mode	CE1#	CE2#	A0	D15:D8	D7:D0
No access (standby mode)	1	1	X	High-Z	High-Z
8/16-bit Mode (even byte)	0	1	0	High-Z	Even Byte
8-bit Mode (odd byte)	0	1	1	High-Z	Odd Byte
16-bit Mode (odd byte only)	1	0	X	Odd Byte	High-Z
16-bit Mode (even & odd byte)	0	0	X	Odd Byte	Even Byte

Since attribute memory contains the CIS, it must be read to determine the card's access timing requirements. However, the speed of the memory device containing the CIS must be known prior to reading it. For this reason, a default access time of 300ns is used for reading attribute memory within all PC cards.

Figure 5-7 shows the typical relationships between signals asserted during attribute memory reads (refer to the PCMCIA standard for minimum and maximum timing values). The HBA starts an access to a given socket when it recognizes an address residing within the PC card. The HBA outputs the target address to the socket at the beginning of the read transfer, along with the REG# signal. The card enable signal, CE1#, is asserted while CE2# remains deasserted, consistent with even byte-only accesses that are permitted to attribute memory. Once the setup time has been satisfied, the read command, OE# (output enable), is asserted, indicating that this is a read from attribute memory. The memory card then returns valid data to the HBA. The HBA keeps the address asserted to the socket to satisfy the recovery (hold) time of the memory device. The PC memory card is now ready for another data transfer.

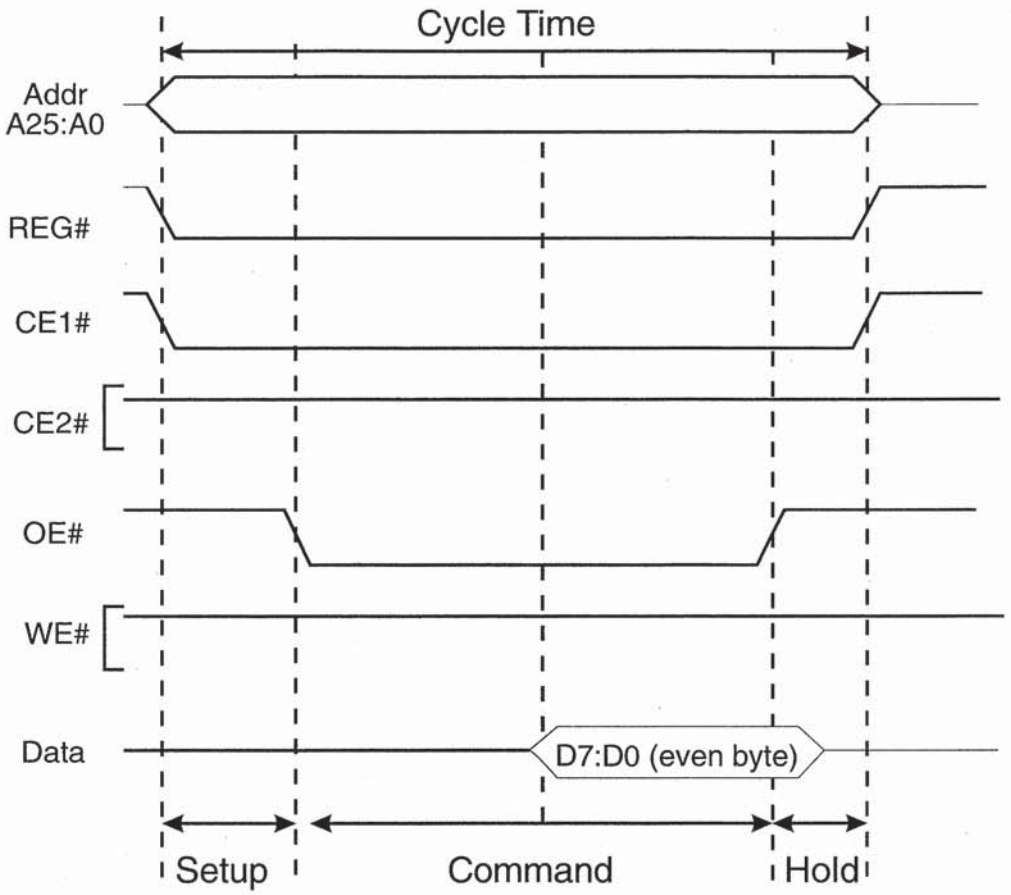


Figure 5-7. Attribute Memory Read Transfer

Attribute Memory Write Transfers

Attribute memory write transfer timing defaults to 250ns cycle timing (consistent with the timing parameters defined for 250ns common memory writes). The only difference between attribute memory write transfers and common memory write transfers is that the REG# signal is asserted during attribute memory writes to distinguish them from common memory writes. Note that the CIS may specify attribute memory write timing, in which case write transfer timing will be determined by the CIS timing entry and not the 250ns default mentioned earlier.

Common Memory Read/Write Transfers

PC Memory cards may be installed in 8-bit host systems or 16-bit host systems. Memory cards respond to two separate address modes, permitting access by either 8-bit or 16-bit host bus systems. The following sections describe accesses by each type of host.

Common Memory Read or Write Transfer (16-Bit Hosts)

The addressing mode used by 16-bit hosts is shaded in table 5-14. Note that 16-bit address mode permits even byte only transfers (over D7:D0), odd byte-only transfers (over D15:D8) and word transfers (over D15:D0). Address lines A25:A1 determine the target word location, while the state of CE1#, CE2# and A0 determines the specific byte or bytes requested within the target word.

Table 5-14. Addressing Mode Supported by PC Memory Card When Connected to 16-Bit Host Systems.

Addressing Mode	CE1 #	CE2 #	A0	D15:D8	D7:D0
No access (standby mode)	1	1	X	High-Z	High-Z
8/16-bit Mode (even byte)	0	1	0	High-Z	Even Byte
8-bit Mode (odd byte)	0	1	1	High-Z	Odd Byte
16-bit Mode (odd byte only)	1	0	X	Odd Byte	High-Z
16-bit Mode (even & odd byte)	0	0	X	Odd Byte	Even Byte

Figure 5-8 illustrates a two byte read from a 16-bit host. Note that REG# is deasserted, indicating an access to common memory. This access results when the host system requests a word from the PC card. The PC card, recognizing that both CE1# and CE2# are asserted, returns an even and an odd byte to the host system. Other combinations of CE1#, CE2# and A0 permit single byte access.

Figure 5-9 shows an example memory write with a 16-bit host system. In this example, an odd byte only is being written.

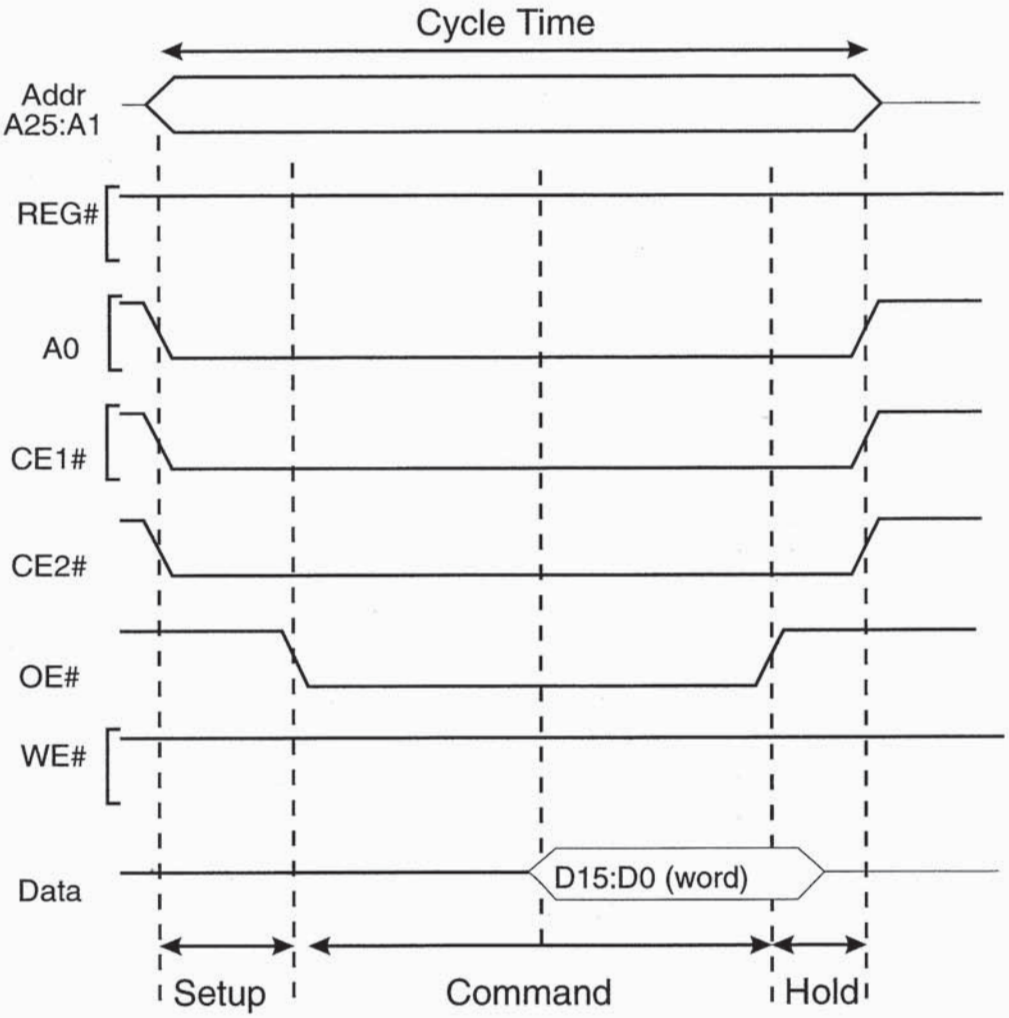


Figure 5-8. Common Memory Read Cycle — Word Transfer

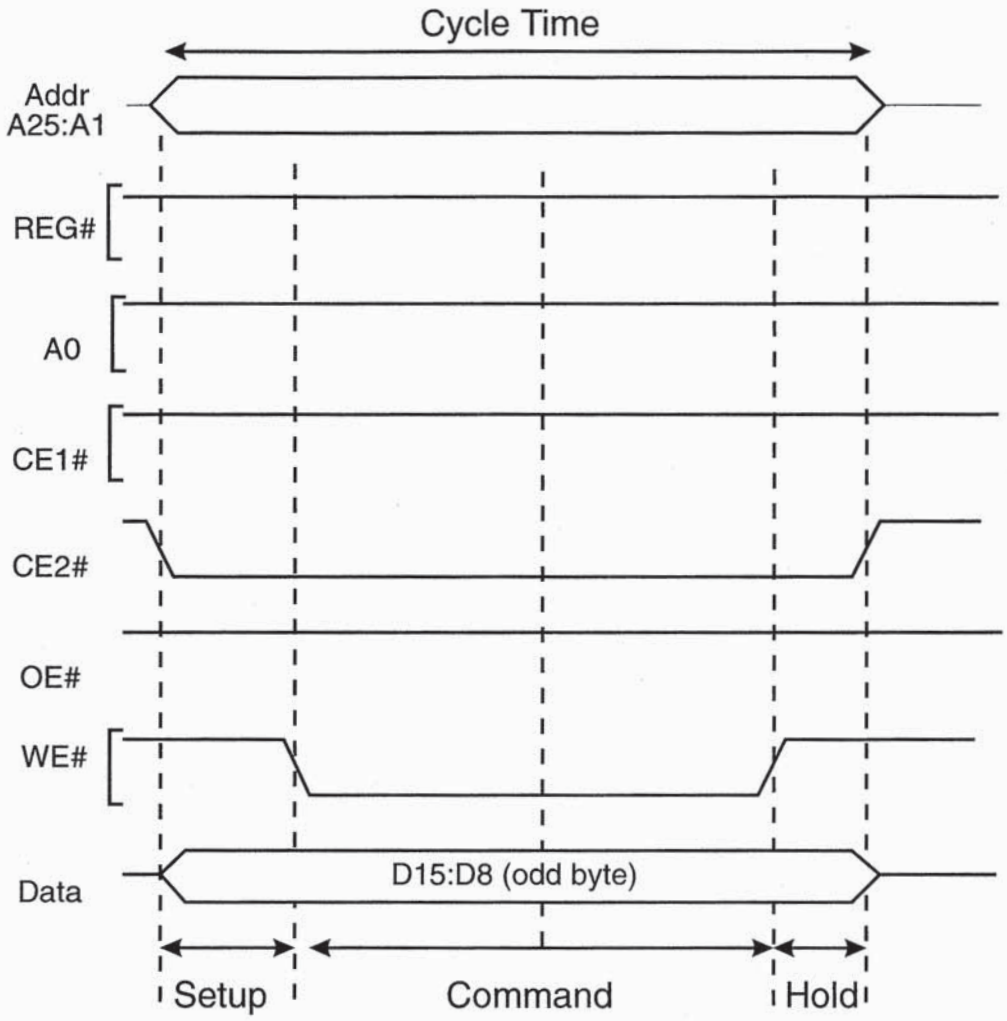


Figure 5-9. Common Memory Write Cycle

Common Memory Read or Write Transfer (8-Bit Hosts)

PC cards must be able to respond to reads and writes from 8-bit hosts that have a single data path (D7:D0). The 8-bit addressing mode permits both even and odd byte accesses over the lower data path, consistent with the needs of the 8-bit expansion bus. As indicated in table 5-15, when the 8-bit addressing mode is used CE2# remains deasserted during all transfers, while CE1# remains asserted. Address A0 specifies whether the access is to the even or odd byte. Note that transfer timing is not affected by the addressing mode used.

Table 5-15. Addressing Mode Supported by PC Memory Card When Connected to 8-Bit Host Systems.

Addressing Mode	CE1#	CE2#	A0	D15:D8	D7:D0
No access (standby mode)	1	1	X	High-Z	High-Z
8/16-bit Mode (even byte)	0	1	0	High-Z	Even Byte
8-bit Mode (odd byte)	0	1	1	High-Z	Odd Byte
16-bit Mode (odd byte only)	1	0	X	Odd Byte	High-Z
16-bit Mode (even & odd byte)	0	0	X	Odd Byte	Even Byte

Common Memory Read/Write Timing with Wait

The WAIT# signal, under PC card control, extends standard cycle time. The maximum duration of WAIT# is 12µs. When WAIT# is asserted by the PC card, command time is extended by the duration of the WAIT# signal. In the event that WAIT# does not extend beyond the standard command time, standard timing will be met. In this instance, timing is not impacted by the assertion of WAIT#.

Figure 5-10 illustrates a memory read transfer with WAIT# asserted. This example illustrates a memory read using 8-bit addressing mode. Note that CE2# is deasserted and CE1# is asserted, while A0 (a logic "1") indicates access to an odd location. Since CE2# is deasserted, the odd location's contents must be transferred via data path D7:D0. The PC card also asserts WAIT#, telling the HBA to extend the cycle time. The cycle completes when WAIT# is deasserted. Refer to the PCMCIA specification for detailed timing information.

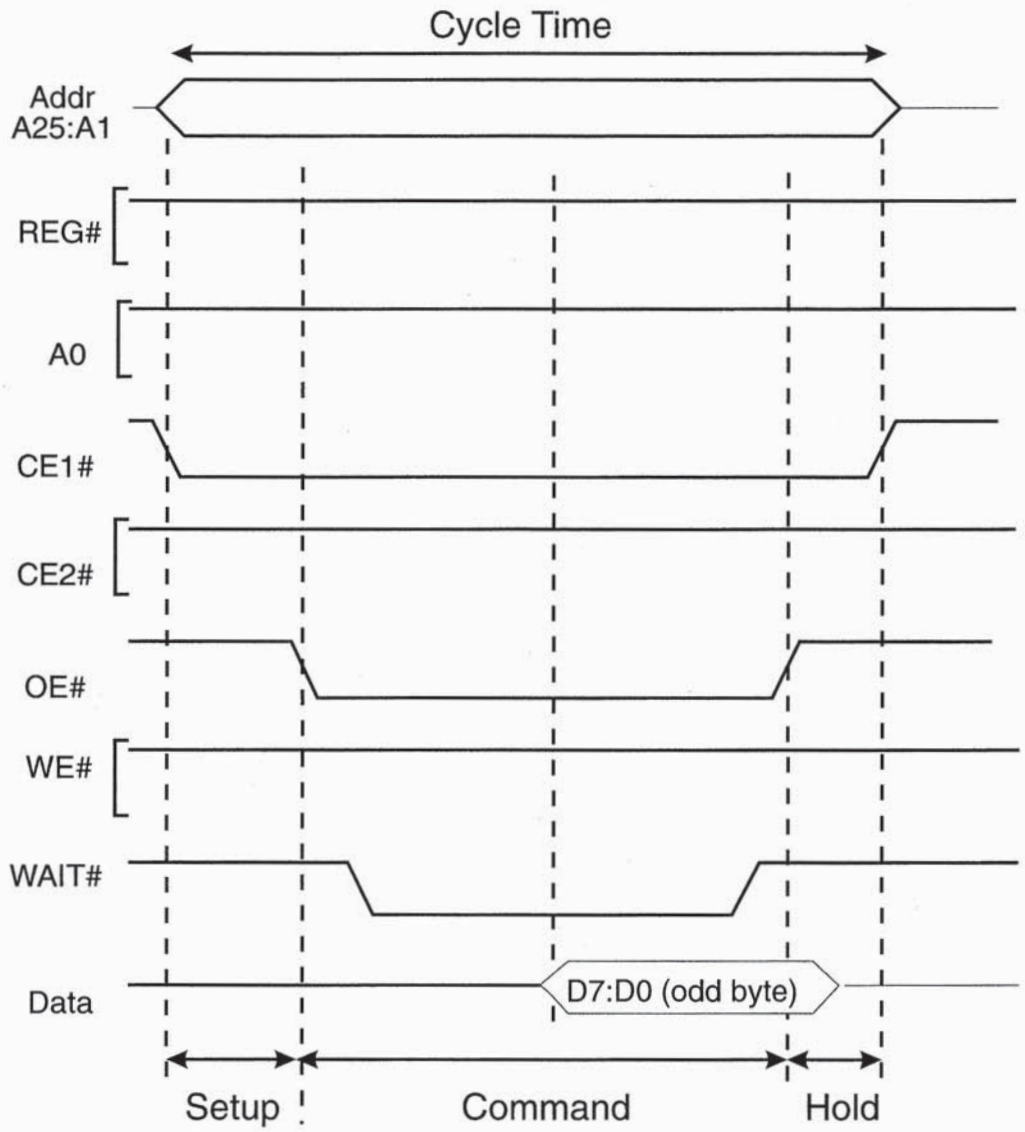


Figure 5-10. Common Memory Read Cycle With Wait Asserted

Chapter 6

The Previous Chapter

The previous chapter detailed the memory-only socket interface. Each pin was defined and its relationship to the PC Card and the HBA was discussed. The memory-only interface is the interface initially seen by 16-bit PC Cards when they are first inserted into a socket. This permits the memory-mapped CIS to be accessed to determine the PC Card type and interface requirements. If the card is designed for an interface type other than memory-only, then the HBA and PC Card are configured to communicate via one of the other interface defined by the PC Card Standard.

This Chapter

This chapter details the memory or I/O socket interface. The memory-only interface is converted into a memory or I/O interface by software after it detects that an I/O PC Card has been installed into the socket interface. Pins that are added or redefined by the memory or I/O interface are discussed along with their relationship to the I/O card function. Some of the memory-only pins are replaced with I/O specific pins when the interface is redefined for I/O. This chapter describes how the functions associated with the replaced memory-only pins are handled.

The Next Chapter

The next chapter defines the electrical interface that permits I/O Cards to use PC compatible DMA transfers. The DMA interface provides a way for standardized I/O cards that use DMA, to take advantage of existing software that is designed to use DMA data transfers.

Overview

The memory-only socket and the Memory or I/O socket have identical pin definitions when a PC Card is initially powered up. That is, when an I/O device is installed in a memory or I/O socket, the pin definition is initially defined as a memory-only configuration, allowing access to the Card Information Structure (CIS), which is mapped in attribute memory address space. Software then reads the CIS, determines the card requires an I/O interface and programs the Host Bus Adapter (HBA) and the I/O card to reconfigure the socket interface to the I/O pin definition.

The memory or I/O interface permits operation of either memory cards, I/O cards or multifunction cards containing combinations of both memory and I/O devices. When a socket contains a memory card, it is defined as a memory-only socket. When an I/O device is installed in the socket, some of the signal definitions change making the socket compatible with both memory and I/O cards.

Since the memory signals are covered in the previous section, this discussion focuses only on the signals that differ from the memory socket implementation.

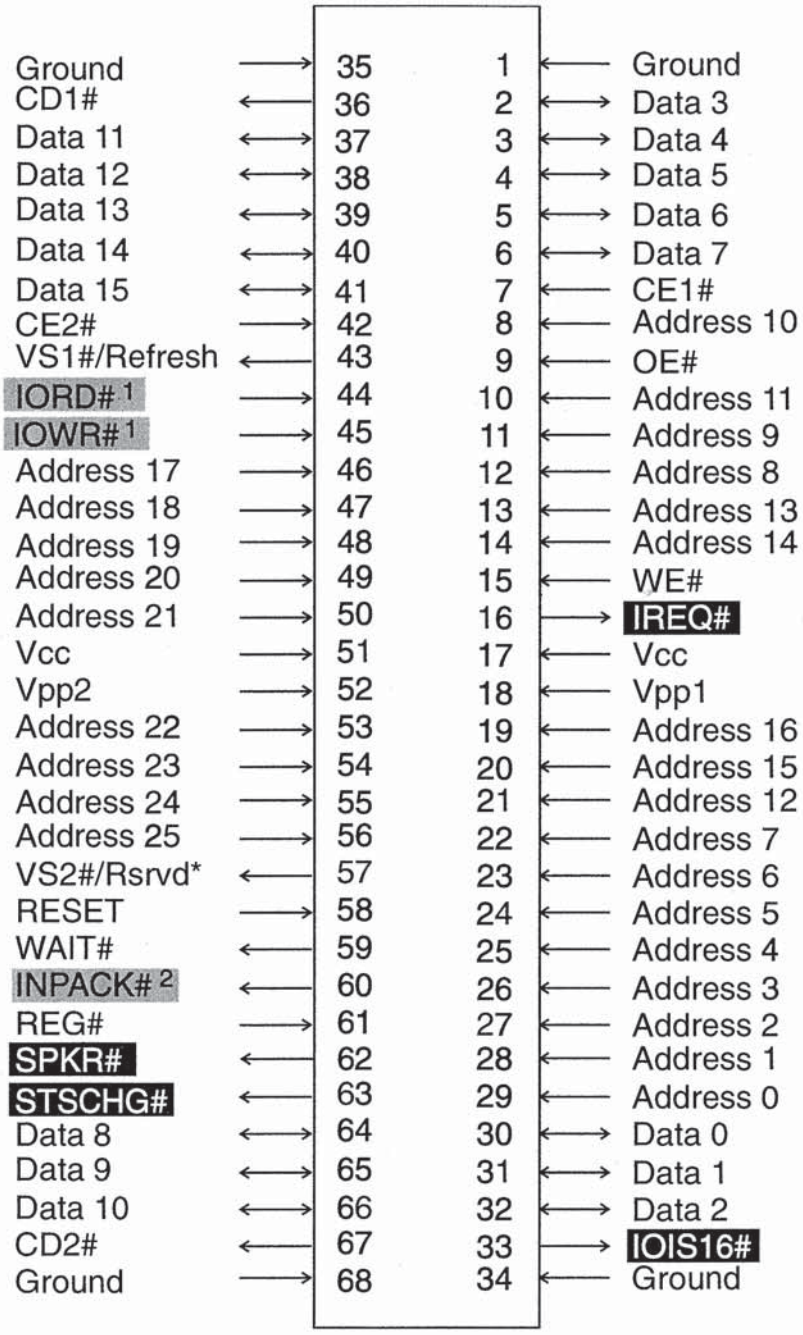
The I/O Socket Interface

The following discussion describes the dynamic changes made to a memory or I/O socket when a PC Card containing I/O devices is installed. Note that the memory or I/O socket is configured as a memory socket when a PC Card is initially installed, allowing the Card Information Structure (CIS) to be read from attribute memory address space. If configuration software detects that the PC Card contains I/O devices, it then programs the PCMCIA HBA to reconfigure the socket for I/O device support.

The signals added to the socket when an I/O device is installed in a memory or I/O socket are listed in table 6-1. Some of these new signals replace pins that are reserved in the memory-only socket (those signals shown in the gray boxes in figure 6-1), while others replace selected status pins used in the memory-only interface (signals shown in the black boxes).

Table 6-1. Pins Added/Removed When Converting from Memory-Only to Memory or I/O Interface

Memory Pin	I/O Replacement	Description
Reserved	INPACK#	**Input Port Acknowledge**. This signal is asserted during I/O read transfers from a PC Card when it recognizes the address. The signal enables the socket's data path transceiver so that the addressed PC Card can deliver valid data to the system. This signal is necessary if addresses within an I/O address window overlap with other devices. (Refer to the chapter entitled "The HBA" for a description of I/O address windows.)
Reserved	IORD#	**I/O Read Command**. This signal is asserted during I/O read transfers from PCMCIA cards.
Reserved	IOWR#	**I/O Write Command**. This signal is asserted during I/O write transfers to PC Cards.
READY	IREQ#	**Interrupt Request**. This signal is asserted to inform the system that the PC Card has an interrupt that needs servicing.
WP	IOIS16#	**I/O size is 16-bits**. This signal is asserted during I/O read transfers from PC Cards, if the device size is 16 bits.
BVD2	SPKR#	**Digital Audio Waveform**. Used to send audio information to the system speaker.
BVD1	STSCHG#	**I/O Status Change**. Used to report a card status change.



1. Pulled-up to Vcc by PC Card (R≥10KΩ). 2. Pulled-up to Vcc by HBA (R≥10KΩ).

Figure 6-1. PCMCIA Memory or I/O Socket

PC Memory or I/O Card Transaction Definition

Six types of transactions can be performed when accessing a PC Card containing I/O devices:

- I/O Read
- I/O Write
- Common Memory Read
- Common Memory Write
- Attribute Memory Read
- Attribute Memory Write

Note that multifunction devices containing both common memory and I/O devices support all the transaction commands.

The Output Enable signal (OE#), Write Enable or Program signal (WE#), I/O Read (IORD#), I/O Write (IOWR#), and Register (REG#) signals define the transaction type. Table 6-2 lists each of the PC Card transaction definition signals and indicates the command signal combinations for common memory, attribute memory and I/O accesses. Note that REG# specifies access to either attribute memory address locations or I/O address locations. The memory and I/O command lines determine which address space is being accessed.

Table 6-2. PCMCIA Transaction Definition

Transaction Type	IORD#	IOWR#	OE#	WE#	REG#
I/O Read	0	1	1	1	0
I/O Write	1	0	1	1	0
Attribute Memory Read	1	1	0	1	0
Attribute Memory Write	1	1	1	0	0
Common Memory Read	1	1	0	1	1
Common Memory Write	1	1	1	0	1

I/O registers incorporated into PC Cards can be either 8-bits or 16-bits wide. I/O cards that contain 16-bit registers assert the IOIS16# signal when a 16-bit register is addressed. The HBA can determine the size of the register being accessed by monitoring the IOIS16# signal and match the bus size to the device size. For example, if a 16-bit access is being made to an 8-bit register, the host system (either the HBA or expansion bus controller) will divide the 16-bit access into two 8-bit accesses required by the I/O device.

The IOIS16# Pin

This pin is asserted by an I/O card when it recognizes that the location being accessed is contained within a 16-bit I/O device. When IOIS16# is asserted the HBA will perform a 16-bit transfer if possible. If the HBA is performed a 16-bit transfer but the IOIS16# pin is deasserted, then the HBA will perform two 8-bit transfers to/from target register.

The IREQ# Pin

The interrupt request pin can be driven either as a level or pulse triggered signal. The interrupt trigger used is a function of the interrupt triggering mechanism used by the host expansion bus. When an interrupt is asserted by the PC Card it is routed (steered) by the HBA to the target interrupt request line on the expansion bus. For details regarding the implementation of the IREQ# trigger selection and interrupt steering refer to the chapter entitled, "The Host Bus Adapter."

The INPACK# Pin

The INPACK# pin is asserted by PC Card I/O functions during reads from registers within the card. This pin is used by the HBA to enable data buffers between the PC Card socket and the expansion bus. If an I/O transfer is sent to a PC Card that does not access an internal register, then the data buffers remain disabled leaving the expansion bus electrically isolated from the PC Card socket interface. This is done in case the I/O access that does not belong to the PC Card is for another device somewhere within the system. Keeping the PC Card isolated from the bus eliminats possible bus loading or data bus contention problems.

The STSCHG# Pin

Many PC Cards that incorporate only an I/O function do not implement memory-mapped devices, require no battery backup, implement no write-protect switch, nor do they have a READY state. Therefore, the status signals (READY, WP, BVD1, and BVD2) which are not defined for the memory or I/O interface are not needed. However, some PC Card's may require one or

more of these status signals (e.g. multifunction PC Cards containing both memory and I/O functions).

Since status signals (READY, WP, BVD1, and BVD2) used to report status information are not defined for the memory or I/O interface, PCMCIA defines an alternative method for reporting these status changes in lieu of the status pins. A configuration register (called the Pin Replacement Register) located within the attribute memory address space is used by PC Cards that must report status for one or more of the status pins that have been removed from the memory or I/O interface.

To notify the HBA that a status change has occurred, the Status Change (STSCHG#) signal is asserted. When the HBA detects the STSCHG# signal asserted it generates a status change interrupt just as if one of the status pins had been asserted on the memory-only interface. Software must then read the pin replacement register contained within the PC Card's attribute memory address space to determine the source of the status change interrupt.

The SPKR# Pin

The speaker pin is used by I/O-based PC Cards to deliver audio information back to the host system's speaker. The signal is implemented as a binary audio signal with a single amplitude, and is simply ORed with other signals that drive the host speaker. If the I/O card does not require a speaker, the SPKR# pin will be driven high by the PC Card.

PC Card functions using the SPKR# pin must also implement an enable/disable bit (called audio enable) in their Configuration and Status register.

I/O Transfers

Cycle times for I/O devices consist of a single timing standard. The default I/O cycle time requires a minimum of 255ns to complete. A device requiring additional cycle time must assert the WAIT# signal to extend the cycle.

Like PC memory cards, PC I/O cards must also respond to 8-bit and 16-bit host addressing modes. These addressing modes are the same as those discussed for memory cards, and are not repeated here.

Unlike PC memory cards, however, I/O devices within PC cards can be designed with either 8-bit or 16-bit registers, or a combination of both. When access is made to an I/O location within the PC card, the HBA may not know whether the I/O transfer is to or from an 8-bit or 16-bit device. In such cases, the HBA samples the IOIS16# signal to determine the size of I/O device accessed. When the host system accesses a PC card's I/O address space, several situations can result that require action:

- Single byte access by the host system to/from 8-bit register
- Word access by the host system to/from 8-bit register
- Odd byte access by the host system to/from 16-bit register
- Word access by the host system to/from 16-bit register

Single Byte Access to/from 8-Bit I/O Devices

Single byte accesses by the host system must be handled so that the data is transferred over the correct data path. Accesses to and from even locations present no problems since both 8-bit and 16-bit hosts expect even address location transfers to occur over the lower data path (D7:D0). Similarly, accesses to and from odd locations present no problems for 8-bit hosts since they expect both even and odd bytes to be transferred over the lower data path. However, 16-bit host systems expect odd location transfers to occur over the upper data path (D15:D8). Since 8-bit I/O devices transfer both even and odd locations over the lower data path, either the host bus system or the PCMCIA HBA must steer the data to the correct path when accessing odd locations.

When an I/O transfer begins, the HBA does not know whether the access is to an 8-bit or 16-bit device, but defaults to the 8-bit addressing mode. This means CE1# is asserted and CE2# is deasserted, while A0 determines whether an even or odd byte location is to be accessed.

Figure 6-2 illustrates a standard I/O read transfer from an odd location. The location being accessed is from an 8-bit I/O register, indicated by IOIS16# being deasserted. Note also that the cycle time is the standard 255ns.

Data from the odd address location is transferred over the lower data path from the 8-bit I/O register. With most PC-based bus architectures, such as ISA, the host system's bus implements device size lines that control data bus steering when accessing odd locations from 8-bit devices. The HBA notifies the host bus controller of the register size being accessed and the bus controller steers the contents of the odd location to the upper data path. If the host

bus does not include a steering mechanism, then the HBA must perform the steering and supply data to the data path expected by the host bus requester.

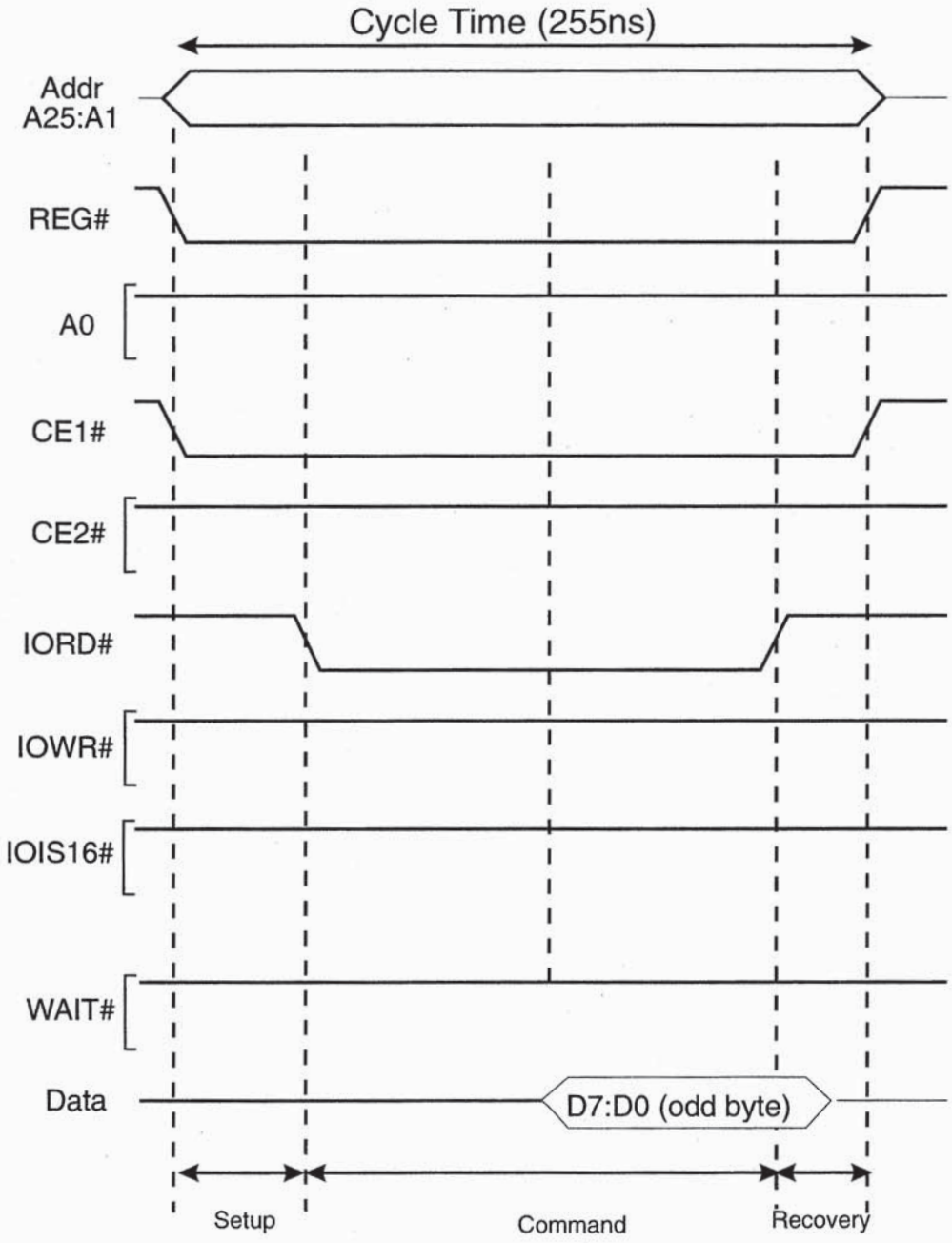


Figure 6-2. Default PC Card I/O Read Cycle.

Word Access to/from 8-Bit I/O Devices

When the host system performs a 16-bit data transfer with 8-bit registers on the PC card, the host's request to transfer an entire word (two bytes) requires two accesses to the PC card. The host system is responsible for converting the word access into two separate byte accesses to the PC card. Depending on the host bus architecture, either the host bus control logic or the PCMCIA HBA must automatically initiate the additional 8-bit access.

Systems based on most PC bus architectures (such as ISA, EISA, and the Micro Channel) employ signal lines and logic designed to manage accesses to and from devices of different sizes. In these instances, the HBA can utilize the host system's resources to run an additional cycle to the 8-bit PC card. When the HBA samples the PC card's IOIS16# line deasserted, it can use the host bus size lines to inform the host that the access is to an 8-bit device. The HBA completes the first byte transfer (even byte) and waits for the second transfer (odd byte) to be initiated by the host. The host, knowing that the odd byte is from an 8-bit device and that the data will be returned on the lower data path (D7:D0), steers the content of the lower data path to the upper path (D15:D8).

Some host systems may not employ device size translation logic for HBAs to use. In these instances, the HBA must run the additional transfer to the PC card and return the entire word requested by the host system in what appears to be a single cycle to the host.

Byte Accesses to/from 16-Bit Register

When the host transfers a single byte (whether with an even or odd location), the transfer can complete in a single cycle when a 16-bit I/O device is accessed. As in the examples discussed earlier, the HBA, not knowing whether the device is 8-bit or 16-bit, starts the transfer by using 8-bit addressing mode. Since even location accesses are identical for both 8-bit and 16-bit addressing mode, no adjustment need be made by the HBA or the PC card and the transfer completes normally regardless of the state of the IOIS16# signal. Transfers to and from odd byte locations when accessing 16-bit devices either require:

- adjustment of the address mode to ensure that data is transferred to and from the PC card over the correct data path, or
- data steering the between upper and lower data paths if 8-bit mode addressing is used.

Word Accesses to/from 16-Bit I/O Registers

Word transfers to and from 16-bit I/O registers can complete in a single cycle. Refer to figure 6-3. Typically, when an I/O transfer begins the address mode defaults to 8-bit addressing mode, in expectation that an 8-bit register may be accessed. However, when the IOIS16# signal is asserted by the 16-bit target device, the HBA can also assert CE2#, causing the PC card to respond to the word transfer.

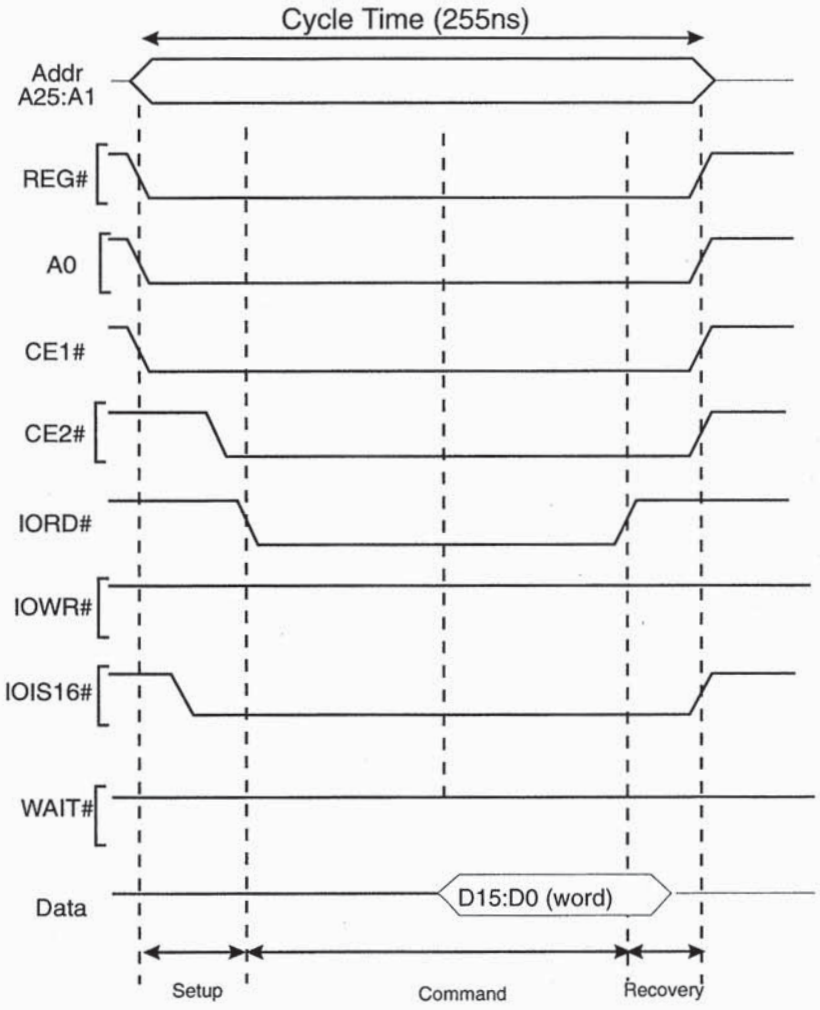


Figure 6-3. I/O Read Cycle without WAIT#- Word Transfer

Chapter 7

The Previous Chapter

The previous chapter detailed the memory or I/O interface between the PC Card and socket. Configuration software reconfigures the memory-only interface into a memory or I/O interface after it detects that an I/O PC Card has been installed into the socket interface. Pins that are added or redefined by the memory or I/O interface are discussed along with their relationship to the I/O card function. Some of the memory-only pins are replaced with I/O specific pins when the interface is redefined for I/O. The previous chapter described how the functions associated with the replaced memory-only pins are handled.

This Chapter

This chapter defines the DMA compliant electrical interface, permitting I/O Cards to use PC compatible DMA transfers. The DMA interface allows I/O devices that use DMA to take advantage of existing compatible software when performing data transfers.

The Next Chapter

The next chapter discusses the PC Card ATA interface. A PC Card ATA interface provides a PC compatible hardware and programming interface that simplifies the job of implemented hard drive solutions in the PCMCIA environment. This chapter defines the various ways that a PC Card ATA can be mapped in the system along with the electrical interfaces that are used. Differences between the PC compatible ATA implementation versus the PC Card ATA interface are also discussed.

Background

Release 2.x systems do not support DMA transfers. As a result, standard PC hardware and software solutions that use DMA will not work with 2.x compliant systems. This makes the job of implementing devices such as floppy drives and Sound Blaster in the PC Card environment extremely difficult. These devices use the IBM PC compatible DMA subsystem to transfer data between themselves and main system memory. Standard software supporting these type devices are designed to program the DMA subsystem in order to transfer data to/from these devices.

Previous implementations of PC Cards that use DMA required definition of additional hardware and software that extends beyond the Release 2.x specification. Some HBA manufacturers included DMA support and some system vendors implemented solutions using these HBAs. These solutions however, did not enjoy industry-wide support because the PCMCIA Standard did not define a standard implementation.

Review of PC Compatible DMA Transfers

PC compatible DMA employs a DMA controller (DMAC) that is programmed to orchestrate a transfer between an I/O device and main system memory. Once programmed, the DMAC handles a block transfer and its termination. The DMAC used in ISA compatible PCs performs a "fly-by" transfer; that is, it is able to transfer data between memory and an I/O device (read and write) in a single DMA cycle without latching the data internally, a job that would normally require two separate bus cycles (a read followed by a write) by the processor.

Upon completion of the overall data transfer, the I/O device interrupts the microprocessor to indicate completion. In response, the microprocessor temporarily suspends its current task and performs an I/O read from the I/O device to check the completion status of the transfer. If the I/O device indicates no errors were encountered, the microprocessor may continue processing.

A DMA Example

This example describes a DMA transfer between the floppy drive controller and system memory. The example defines the signal involved in the transfer and describes the sequence of events that occur to initiate, perform, and terminate a DMA transfer.

Four distinct steps must occur to initiate and complete a DMA transfer:

- Set-up the DMA channel for the transfer.
- Command the I/O device to initiate the block data transfer.
- Grant the system buses to the DMA controller so it can run the bus cycle.
- Notify the microprocessor that the transfer is complete.

Each DMA Channel within the DMA controller has its own set of I/O registers that the programmer uses to set up the data transfer. The set of I/O registers associated with each DMA Channel allows the programmer to specify:

- The Transfer Count (number of bytes to be transferred).
- The start memory address (start address in memory where data will be read from or written to).
- The direction of transfer with reference to (type of transfer).

After the DMA Channel has been set up by the programmer, the I/O device must be programmed to initiate the overall block data transfer. As an example, the programmer would issue the proper series of I/O write commands to a floppy disk controller to initiate a disk read operation.

Having set up the respective DMA channel and issued the proper commands to the I/O device (in this case, the floppy disk controller and DMA channel two), the microprocessor can then go on to another task. The entire data transfer and its termination will be handled by the I/O device and its respective DMA Channel.

Each I/O device designed to use the DMA transfers employs three dedicated DMA signals defined by the ISA bus:

- **DREQ** (DMA Request) — output by the I/O device to request a DMAC transfer be performed.

- **DACK#** (DMA acknowledge) — input to the I/O device notifying it that the transfer has started.
- **TC** (Terminal Count) — input to the I/O device notifying it that the block transfer has been completed.

When performing a floppy disk transfer, the programmer may only perform data transfers in multiples of the sector size. When running MS-DOS, a sector on a floppy disk contains 512 bytes of information. This would be the smallest data transfer possible when transferring information between a disk drive and memory.

Refer to figure 7-1 for a block diagram of the components involved.

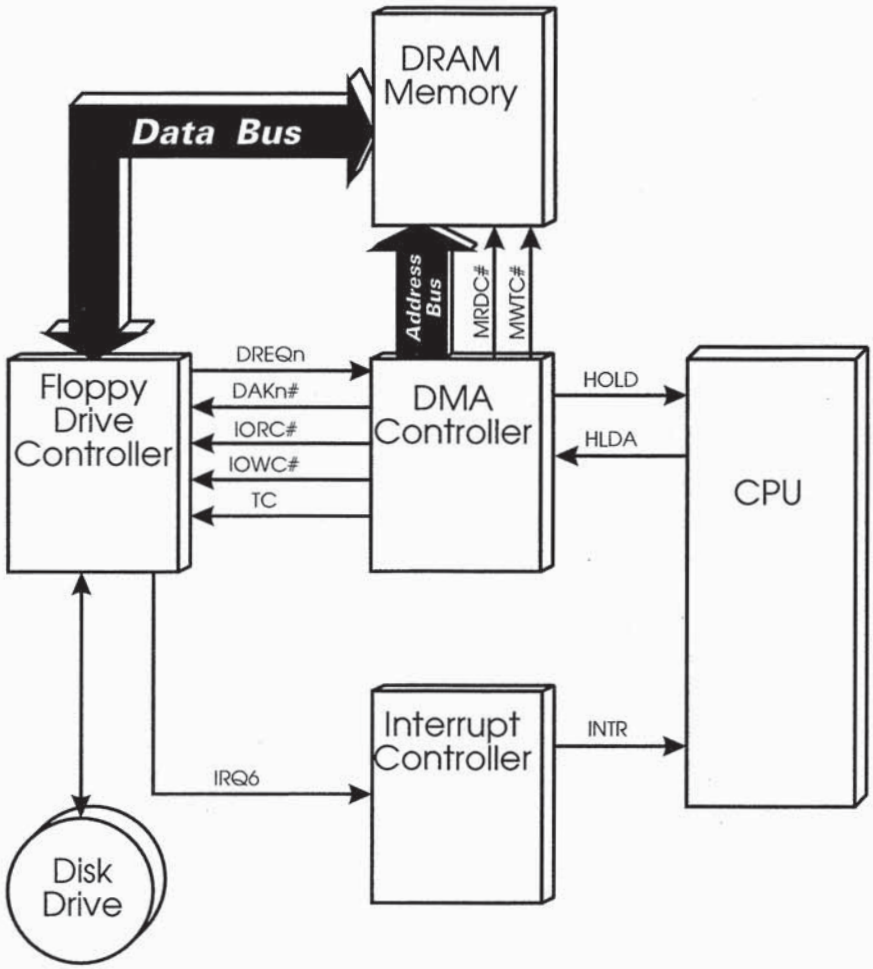


Figure 7-1. Example DMA Transfer Mechanism

1. The programmer issues a series of I/O writes to DMA channel two's registers to set up the start memory address, transfer count and the direction of the transfer with reference to memory. In the disk read example, assume that the start memory address is 1000h, the transfer count is 512 bytes (one sector) and that this is a write transfer (with reference to memory).
2. The floppy controller must be programmed with the disk read command, the cylinder number, the head (or surface) number, the start sector number and the number of sectors to be read. In this example, we will assume a one sector (512 bytes) read from cylinder three, head one, sector five.
3. Upon receiving the parameters and the disk read command, the disk controller initiates a seek operation to position the read/write head mechanism over cylinder three. It must then wait for the disk to spin until the start of sector five is detected under the read head on surface one. As the disk is a mechanical device, this will take some time.
4. In the example of a read operation from a floppy disk controller, some time will elapse before the disk controller has read the first byte from disk.
5. When the first byte has been transferred from the disk to the disk controller, the floppy disk controller must then request that its associated DMA Channel transfer the data. The floppy disk controller asserts DMA request line two (DREQ2) that goes to the DMAC.
6. The DMAC responds by asserting HOLD to seize the buses (address, data, and control buses) from the microprocessor. The HOLD line goes directly to the microprocessor's hold request input.
7. When the microprocessor completes the current bus cycle, it will tri-state all of its bus output drivers, thereby floating the buses. The microprocessor also asserts HLDA to tell the requesting device (the DMAC in this case) that it is now the bus master.
8. The DMAC then responds to DREQ2 from the disk controller by activating DMA Acknowledge (DACK2#) and the I/O read command line (IORC#). These two lines go to the disk controller.
9. The disk controller drops DREQ2 and begins the access to the data register to place data onto the data bus.
10. The DMAC activates the Memory Write Command line (MWTC#) and places the address from channel two's start address register onto the address bus.
11. The data on the data bus is written into memory at the address currently on the address bus.
12. The DMAC then increments channel two's memory address register by one to point to the address in the RAM where it will store the next byte it receives from the disk controller.

13. The DMAC also decrements the byte transfer count. If the transfer count is not exhausted, the data transfer is not complete and the DMAC must wait for another DMA Request (DREQ2) from the floppy drive controller, indicating that it has another byte to transfer. The DMAC also deasserts HOLD to give control of the buses back to the processor and awaits the next DREQ2 assertion from the floppy disk controller. The microprocessor reattaches itself to the buses (exits the tri-state condition) and deasserts the Hold Acknowledge line (HLDA) to tell the DMAC it has resumed control of the buses. The microprocessor is now bus master again. (step 6).
14. When the transfer count is exhausted, the data transfer is complete. The DMAC will once again deassert the hold request line (HOLD) to tell the microprocessor that it no longer needs the buses.
15. The DMAC also generates EOP (End-of-Process). This supplies the signal TC (Terminal Count reached) to the disk controller. The disk controller will then generate a device-specific interrupt request to the 8259 Interrupt Controller which in turn generates INTR (Interrupt Request) to the microprocessor to inform it that the transfer operation is complete.

The DMA controller supports a variety of transfer modes. This example describes a transfer in which the DMAC surrenders control of the buses after each transfer, called Single Transfer Mode. This transfer mode keeps the processor and other bus masters (e.g. the refresh logic) from be starved for control of the system bus. Other transfer modes are supported by the DMAC including Block Transfer Mode and Demand Transfer Mode.

DMA Channels Supported by ISA

The DMAC used in ISA machines is the Intel 8237, providing four separate DMA channels. Two 8237s are used in a master/slave configuration, providing a total of seven DMA channels. Each DMA channel employs its respective DREQn and DACKn# signals (where n= 0, 1, 2, 3, 5, 6, 7), corresponding to the DMA channel number. Each DMA channel is used by a separate I/O device to handle block data transfers with memory.

ISA compatible machines support both 8- and 16-bit DMA channels, specifying the width of the data path used during the transfer. DMA channels 0-3 are 8-bit only DMA channels and DMA channels 5-7 are 16-bit only DMA channels. An I/O device wishing to use DMA transfers must select one of the DMA channels corresponding to the width of transfer it supports.

For more detailed information regarding DMA refer to the MindShare book entitled, *"ISA System Architecture,"* published by Addison-Wesley.

The DMA Socket Interface

The DMA interface defines three DMA signals (DREQ#, DACK, and TC) required by an I/O device in order for it to use the DMA transfer mechanism. Figure 7-2 illustrates the memory or I/O interface signals that can be reassigned to the respective DMA related signals.

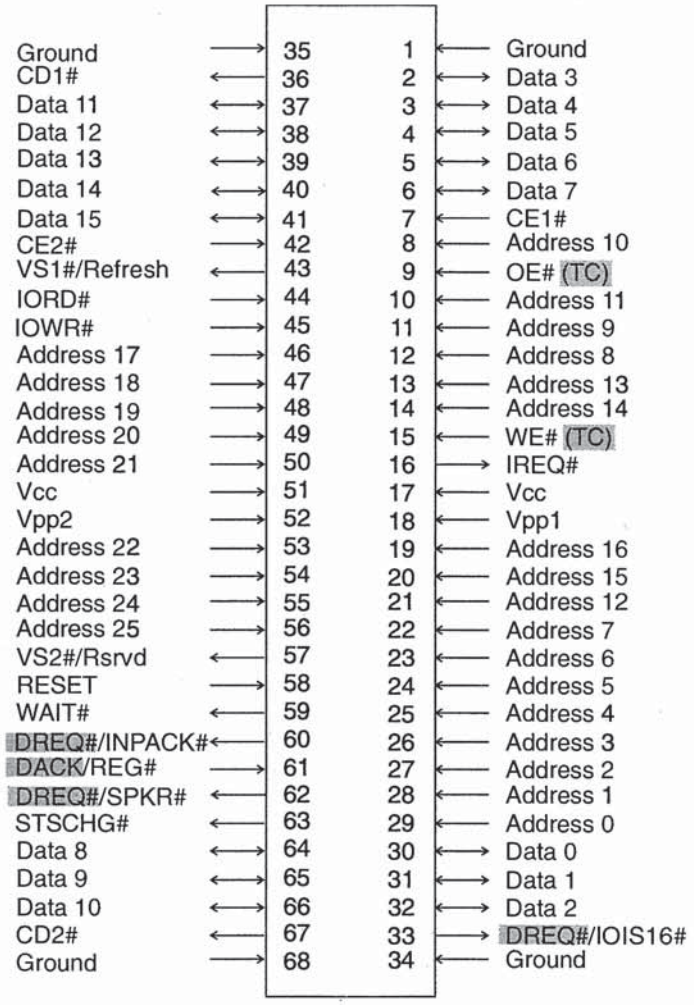


Figure 7-2. DMA Signal Interface

The DREQ# Pin

DREQ# can be assigned to any one of three memory or I/O interface pins, thereby replacing the standard I/O signals as follows:

- Pin 33 (replaces IOIS16#)
- Pin 60 (replaces INPACK#)
- Pin 62 (replaces SPKR#)

The pin assigned to fulfill the DREQ# signaling function is specified by the PC Card's CIS. This DREQ# pin assignment information is contained within the miscellaneous features field of the Configuration Table Entry tuple (tuple code 1Bh). The miscellaneous features field definition is illustrated in table 7-1.

Table 7-1. Definition of the Miscellaneous Features Field that Defines DMA support

7	6	5	4	3	2	1	0
EXT	RFU (0)	Pwr Dn	Read Only	Audio	Max Twin Cards		
EXT	RFU (0)	RFU (0)	DMA width	DMA Request Signal		RFU (0)	RFU (0)

The first byte within the miscellaneous features field was defined for 2.x compliant systems and contains no definition for DMA. The newer PC Card Standard defines an extension that includes the DMA related information. Bit 7 of the first miscellaneous features extension byte is set to indicate that the second byte is present. Bits 2 and 3 of the second byte specifies which interface pin the PC Card uses for DREQ#. A PC Card that uses DMA forfeits the functionality of the signal replaced by DREQ#. The binary values in bits 2 and 3 are interpreted as shown below:

Table 7-2. Interpretation of DMA Request Assignment Bits

Bit 3	Bit 2	Definition
0	0	DMA not supported
0	1	DREQ# uses SPKR#
1	0	DREQ# uses IOIS16#
1	1	DREQ# uses INPACK#

Note that the data width of the DMA transfer is also specified by bit 4 of the second miscellaneous features byte. A value of "0" indicates an 8-bit DMA

data transfer is supported and a value of "1" indicates support for 16-bit DMA transfers. Configuration software is responsible for checking the DMA data width and selecting the corresponding DMA channel that supports the data width indicated by the PC Card.

The DACK/REG# Pin

The DACK is shared with the REG# pin to differentiate normal I/O transfers from DMA transfers. When an I/O register access begins the HBA asserts either IORD# or IOWR#, signaling the start of the transfer. It further specifies the type of I/O access using the DACK/REG# pin. The access is treated as a normal I/O transfer when REG# asserted (in which case the PC Card decodes the address to select the target register) and when DACK is asserted the transfer is recognized as acknowledgment of a DMA transfer (in which case the DMA data register is selected directly and the address is ignored).

The TC Pin

When TC is asserted by the DMA controller, it is indicating that the block transfer has completed. Note that only one TC pin is defined by the ISA expansion bus, therefore the DMA controller also asserts the respective DACKn# signal to identify the I/O device that TC is intended for. The PC Card upon recognizing TC belongs to it, asserts its IREQ# pin, thereby signaling the system that its DMA transfer has ended.

TC is assigned to pin 9 (replacing OE#) to specify completion of a DMA write transfer and pin 15 (replacing WE#) to specify completion of a DMA read transfer.

DMA with PC Card

Figure 7-3 illustrates a PC Card using DMA transfers. Notice that the HBA must invert the DREQ# and DACK signals to/from the ISA bus, since it defines these signal active in the opposite logic state. DMA transfers complete just as they do to/from any I/O device supporting the DMA transfer mechanism. Additionally the HBA must implement DMA channel steering logic so that the PC Card's DREQ# and DACK signals connect to the selected ISA DREQn and DACKn# signals.

DMA transfer width may be either 8-bit or 16-bit as specified in the miscellaneous features field of the Configuration Table Entry tuple (table 7-1). The HBA determines the DMA transfer size (8- or 16-bit) via the SA0 and SBHE# signals on the ISA bus and in turn asserts either CE1# (8-bit transfer) or CE1# and CE2# (16-bit transfer). Refer to the chapter entitled, The "Host Bus Adapter" for additional information regarding the DMA implementation.

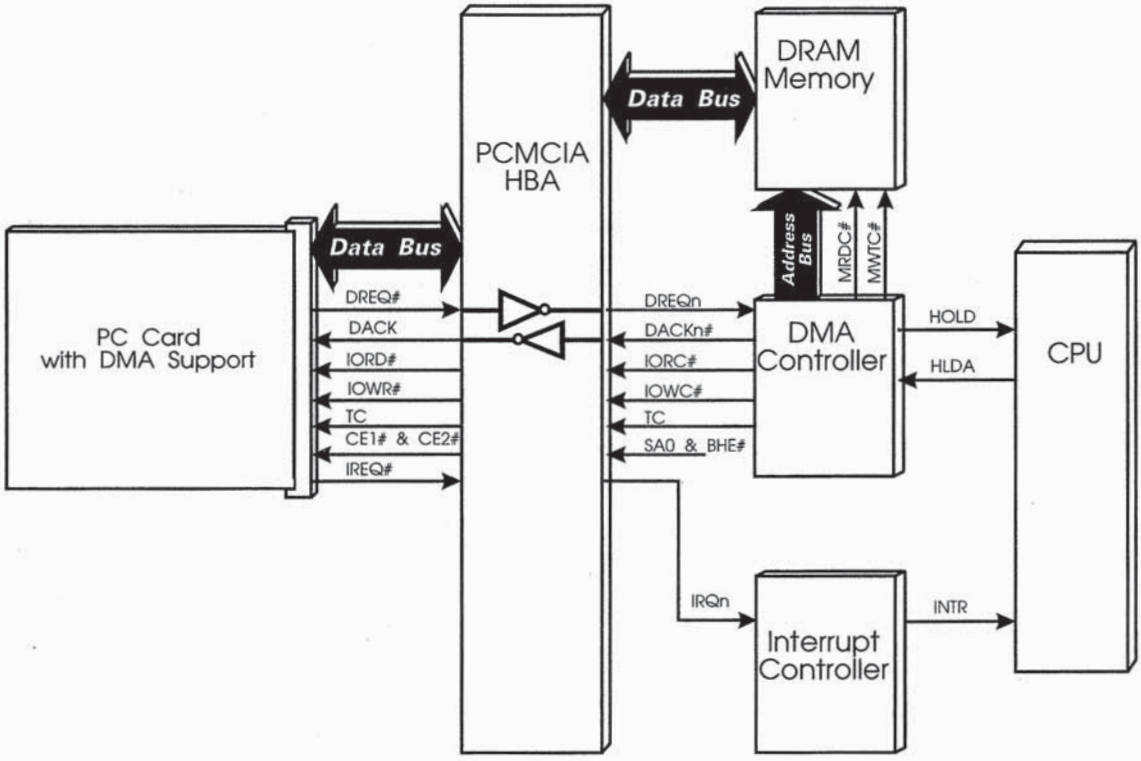


Figure 7-3. Block Diagram of PC Card implementing DMA Transfers

DMA Transfer Timing (PC Compatible)

DMA Bus Cycle

In the PC environment, DMA transactions are performed by an Intel compatible 8237 DMA controller (DMAC). The DMAC transactions consist of four states, each with a duration of one DMA clock period. When the DMAC has

gained ownership of the system buses, it uses its own clock when executing bus cycles. This clock is referred to as the DMA clock and is 1/2 the expansion bus clock frequency. Depending on the PC and the selected processor speed, this will yield a DMA clock of either 3MHz (6MHz AT), 4MHz (8MHz ISA-compatible machine), or 4.165MHz (8.33MHz ISA-compatible machine).

Table 7-3 lists the clock period for the three possible processor speed settings:

Table 7-3. Typical DMA Clock Speeds in the PC Environment

Speed Setting	DMA Clock Frequency	DMA Clock Period
6MHz	3MHz	333.3ns
8MHz	4MHz	250ns
8.33MHz	4.165MHz	240ns

Prior to receiving a DMA Request, the DMAC is in the idle state, Si. When a DRQ is sensed, the DMAC enters a state where it asserts HOLD (Hold Request) to the microprocessor and awaits the HLDA (Hold Acknowledge). This state is called So. The DMAC remains in the So state until HLDA is sensed active.

The DMAC can then proceed with the DMA transfer. S1, S2, S3 and S4 are the states used to execute a transfer (of a byte or word) between the requesting I/O device and system memory. In addition, when accessing a device that is slow to respond, a DMA transfer cycle can be stretched by deasserting the DMAC's READY input until the device is ready to complete the transfer. This will cause the DMAC to insert wait states, Sw, in the bus cycle until READY goes active again.

The following actions take place during states S1-S4. See figure 7-4 for the actual timing of a single transfer:

State	Actions Taken
S1	During single transfer mode, S1 is used to output the middle byte of the memory address, A8:A15 during each transfer. The middle byte of the memory address is output onto data bus pins D0:D7. The DMAC also pulses its Address Strobe (ADSTB) output during S1 and on the falling edge of ADSTB the new middle byte of the address to be latched into the external DMA address latch. Address lines A8:A15 receive the middle portion of the address during S2. During Block and Demand transfers, the middle byte of the memory address only changes once every 256th transfer. For this reason, when in these modes the DMAC only enters the S1 state every 256th transfer to update the middle byte of the address.
S2	During S2, the lower byte (A0:A7) of the memory address is output directly onto the address bus, A0:A7. The DMAC's AEN output is set active causing the external DMA address latch to output the middle portion of the address and to act as an enable for the DMA Page Register addressing. In addition, DACKn# is asserted to tell the I/O device that the transfer is in progress. When DACK# is asserted, the HBA starts the PC Card data register access.
S3	S3 will only occur in a bus cycle if Compressed Timing hasn't been selected for this DMA channel. See text below for a discussion of Compressed Timing. During S3, the MRDC# or the IORC# line is set active. If the DMA channel is programmed for extended writes, the MWTC# or IOWC# line is also set active during S3.
S4	If the DMA channel was not programmed for extended write, the MWTC# or IOWC# is set active at the start of S4. If extended write had been selected, the write command line was already set active at the start of S3. The actual read/write takes place at the trailing edge of S4 when both the Read and Write command lines are de-asserted by the DMAC. This completes the transfer of a byte or word between memory and the requesting I/O device.

When Compressed Timing is selected, S3 is eliminated from the DMA transfer cycle. The only real purpose of S3 is to allow the Read command line to be asserted for twice the duration it is when Compressed Timing is active. Not all memory and I/O devices will tolerate this abbreviated Read command line, so it must be used cautiously.

When Extended Write is selected, it causes the Write command line to be set active during S3 rather than S4, effectively doubling the duration of the Write command line's active period.

It should be obvious that Extended Write and Compressed Timing are mutually exclusive because S3 is essential for Extended Write and is eliminated when Compressed Timing is selected.

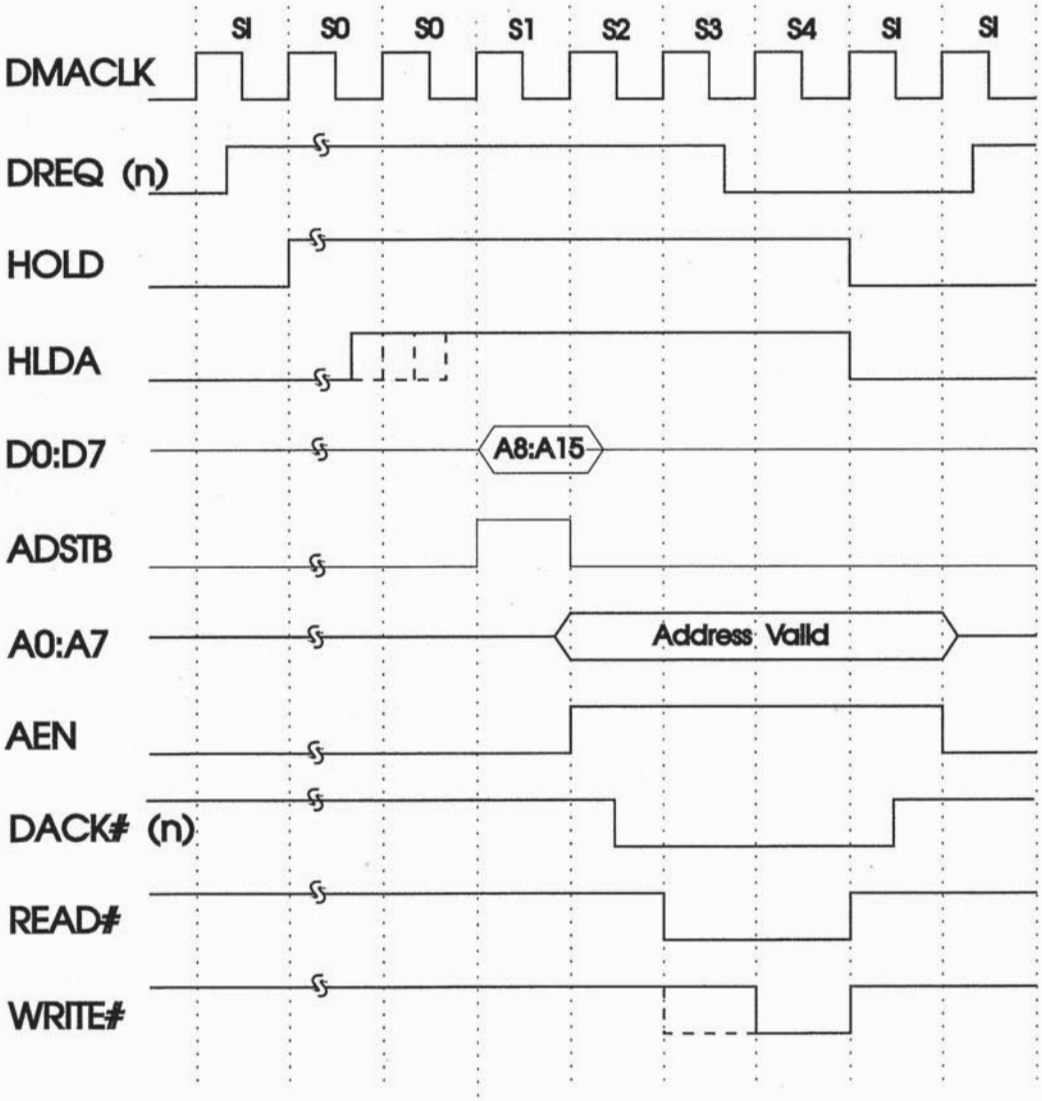


Figure 7-4. Single Transfer Mode Timing

The HBA must satisfy the timing requirements of the DMA controller implemented in the host system. Note that the duration of the PC Card access is during the assertion of the DACK# from the ISA bus (DMA clocks S2, S3 and S4).

Changes to Socket Services

Several socket service functions have been modified to support DMA. These functions are listed in table 7-4 and the DMA change is described.

Table 7-4. Socket Service Functions Modified to Support DMA

Socket Service Function	Code	Description of Change
GetSocket	8Dh	Provides status information regarding the current setting of the specified HBA socket interface. Includes two bits that specify whether the DREQ# signal is currently assigned to the socket (00b=DREQ# not assigned to socket), and if so, which socket interface pin is used to signal DREQ#. (01b=SPKR#; 10b=IOIS16; 11b=INPACK)
InquireSocket	8Ch	Defines attribute bits indicating whether the socket supports DMA transfers. If supported, also defines a bit-map of DMA channels supported.
SetSocket	8Eh	Changes the current settings of the HBA socket interface. Includes the same definition used by Get socket
GetStatus	8Fh	Provide status regarding the HBA socket interface and PC Card configuration. This information incorporates the changes made to GetSocket.

Changes to Card Services

Several of the card services have also been modified to support DMA. The services affected are listed in table 7-5 along with a description of the changes made. The services shown in shaded boxes were added with the PC Card 95 release.

Table 7-5. Modifications Made to Card Services to Support DMA

Service Name	Code	Description of Change
AdjustResouceInfo	35h	Obtains status of or makes changes to the system resource table managed by card services. If DMA is supported, DMA channel resources will be included in the resources table, and the ability to modify DMA resource information within the table will be defined for this service.
GetConfigurationInfo	04h	Provides the current configuration of the PC Card and socket. This service includes DMA configuration information.
ModifyConfiguration	27h	Modifies the current configuration of the socket configuration. Adds support for enabling or disabling the DMA channel.
RequestConfiguration	30h	Requests that card services perform the configuration that has been specified. This service adds the ability to enable or disable the DMA routing, if a Request-DMA service was called for this PC Card.
RequestDMA	3Ah	Requests that a DMA channel be assigned to the PC Card being configured.
ReleaseDMA	3Bh	Releases a DMA channel previously assigned to a PC Card with the RequestDMA service.

Chapter 8

The Previous Chapter

The previous chapter defined the DMA compliant electrical interface, permitting I/O Cards to use PC compatible DMA transfers. The DMA interface allows PC Cards using DMA to take advantage of existing compatible software when performing data transfers.

This Chapter

This chapter discusses the PC Card ATA interface. A PC Card ATA interface provides a PC compatible hardware and programming interface that simplifies the job of implemented hard drive solutions in the PCMCIA environment. This chapter defines the various ways that a PC Card ATA can be mapped in the system along with the electrical interfaces that are used. Differences between the PC compatible ATA implementation versus the PC Card ATA interface are also discussed.

The Next Chapter

The next chapter focuses on the optional Auto-Indexing Mass Storage (AIMS) interface. The chapter describes the programming interface and the transfer mechanism used by AIMS cards.

The ATA Interface

IDE Disk drives use the ATA (AT Attachment) interface common in many ISA-compatible PCs. The ATA interface is a 40 pin interface that connects IDE drives to an ATA host bus adapter (HBA) interface (refer to figure 8-1). The interface includes register select signals, three address lines, 16 data lines and control signals. Functions performed by the ATA host adapter include address decode and data buffering.

Devices implemented as ATA drives in the PCMCIA environment include actual small form-factor disk drives (i.e. 1.8 inch disks) and Flash cards that use the ATA interface and emulate a disk drive. The ATA interface is attractive because the software interface used by ATA devices is standardized and quite common. Standard BIOS routines are built into virtually every PC to support the ATA interface, eliminating the need for specialized device driver code to access the ATA card.

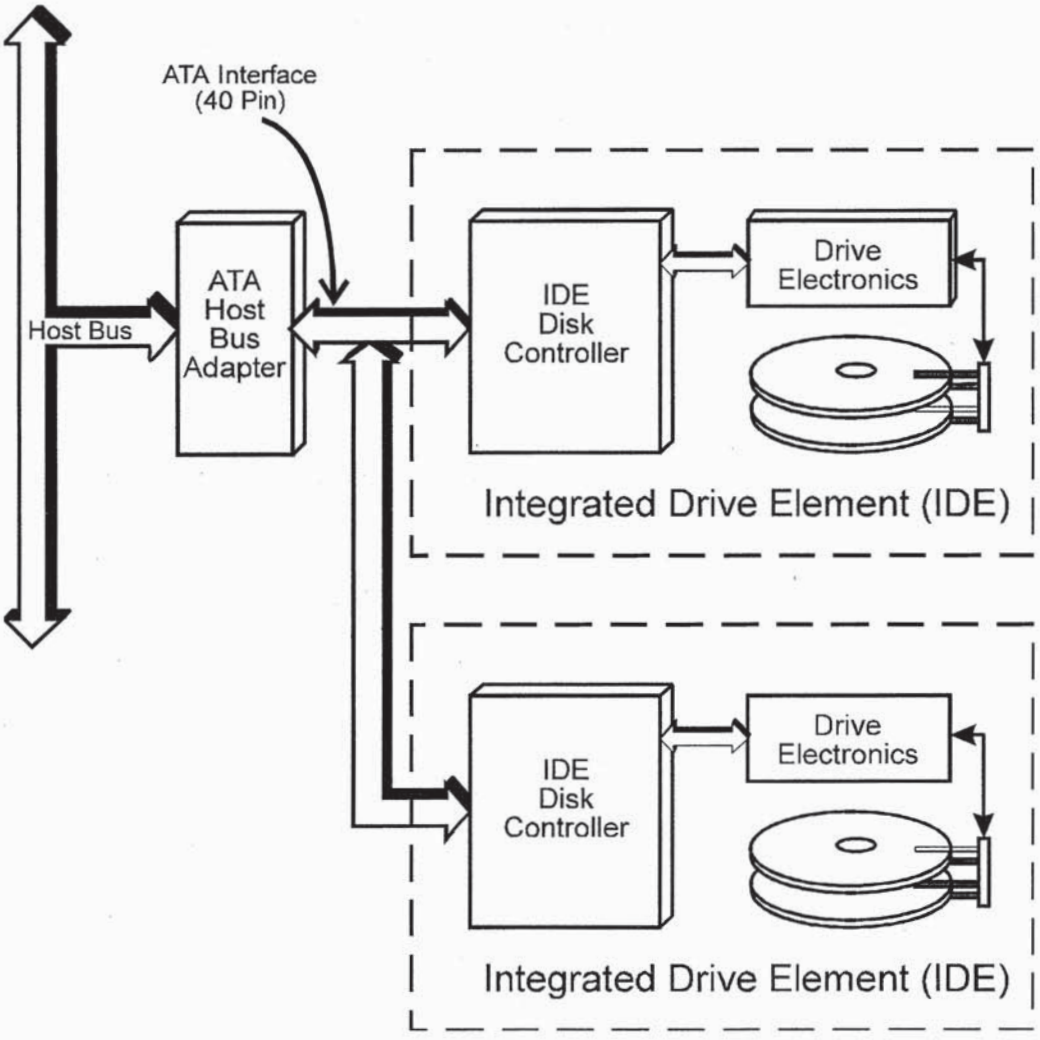


Figure 8-1. Typical ATA Interface to IDE Drive

The ATA Interface

PCMCIA does not require that the HBA include the ATA host adapter functionality. In the PCMCIA environment, the ATA host adapter functions (address decode and data buffering) are incorporated into the ATA PC Card proper as shown in figure 8-2. As a result, PC Card ATA devices can interface via the standard I/O socket interface.

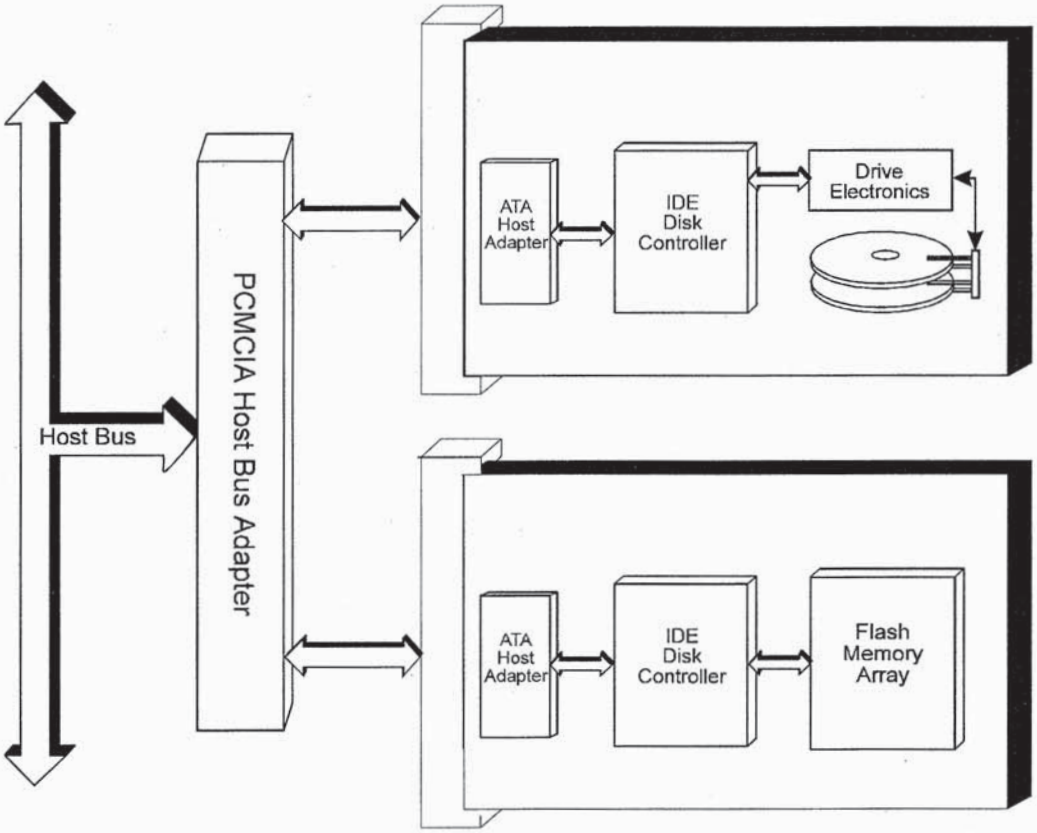


Figure 8-2. PC Card ATA Disk and Memory Devices

The PCMCIA specification includes an interface definition, which lists the minimum signals required to communicate with an ATA PC Card. However, virtually all PCMCIA host bus adapters support the standard memory or I/O interface, which contains all the signals required to support the ATA interface. Figure 8-3 illustrates the minimal interface required to support ATA Cards. Notice that all of the signals defined are included in the standard memory or I/O socket interface.

It should be noted that some PCMCIA host bus adapters provide an ATA interface (defines the standard 40 pin interface required by IDE drives) at the PC Card socket. This interface is not specified nor supported by PCMCIA. These designs integrate the ATA host bus adapter into the PCMCIA HBA. This interface permits system manufacturers to connect a 40 pin standard small form factor IDE drive via the PC Card socket. One advantage of the small form factor ATA drive interface is that it can provide performance advantages over the standard I/O interface that requires the ATA host bus adapter be integrated into the PC Card itself. This performance gain results from the elimination of address setup time (required by standard PC Cards) when transferring data to and from the ATA drive. See the chapter entitled, "An Example Adapter - CL-PD6722."

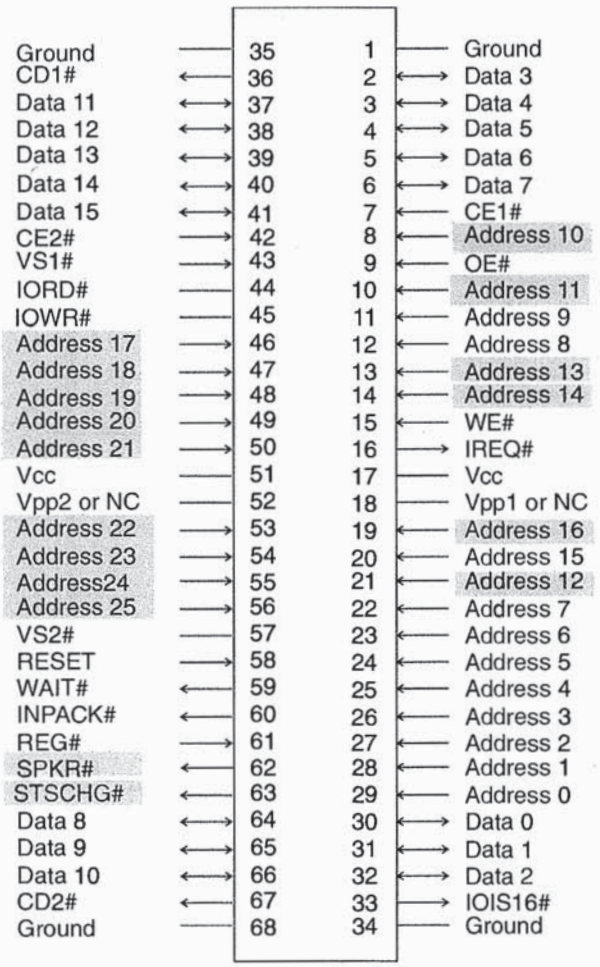


Figure 8-3. Minimum Signals Required for ATA Socket Interface

Differences Between Standard ATA and PCMCIA ATA

Note that the ATA interface consists of signals and functionality not completely supported by the PCMCIA ATA interface. Note for example, that the normal ATA interface supports daisy chaining another drive via a ribbon cable connector. Since PC Card ATA drives have the ATA adapter integrated into the PC Card, the daisy chain approach is not supported with the PCMCIA solution. Table 8-1 lists the ATA signals not defined by PCMCIA's ATA interface and identifies the functionality that may be impacted or lost.

Table 8-1. Signals Defined by ATA But Not Used By PCMCIA

ATA Signals Not Supported	Related Functionality
PDIAG (Passed Diagnostics)	Asserted by drive 1 to inform drive 0 that it has completed diagnostics. This pin allows drive 0 to report diagnostic status for drive 1. The diagnostic command must be run for each socket containing an ATA PC Card.
DASP (Drive Active/Second Drive Present)	This is a multiplexed signal indicating that a drive is active. During initialization, this signal is asserted by drive 1, indicating its presence. This functionality is provided by PCMCIA using the Socket and Copy Registers.
CSEL (Cable Select)	Used by the ATA host adapter to select drive 0 or drive 1. PCMCIA Drives use the Socket and Copy Registers to differentiate between drive 0 and 1. Each are accessed at a separate socket based on the copy number.
DMARQ and -DMACK (DMA Request and DMA Acknowledge)	These signals permit ATA drives to use DMA transfers when transferring data to and from the 16-bit data register. This capability is not supported by PCMCIA.

ATA System Resource Requirements

ATA devices contain two register blocks called the command register block and control register block. Each of these register blocks must be assessible by the system. PC Card ATA devices support either I/O or memory-mapping these registers using one of four addressing modes listed in table 8-2.

Standard mapping in the ISA environment includes the assignment of two separate I/O address ranges to map ATA drive registers into. If these ranges are not available, another range of I/O addresses can be used. If neither of the standard I/O address ranges are available, then a contiguous block of 16 I/O

locations is acquired for mapping the command and control block registers into.

Alternatively, the registers can be mapped into memory locations. When memory-mapping is chosen, a contiguous 2KB block of memory locations are used. The command and control registers are mapped into the first 16 bytes of the 2KB memory block, while the last 1KB of the block is used as a high speed buffer to transfer data to and from the PC card. The address modes listed in table 8-2 (except for the memory-mapped option) are mandatory for compliance with the PCMCIA ATA specification.

Table 8-2. ATA Addressing Options Supported by PCMCIA

Address Mode	Command Block	Control Block
I/O - Primary ATA drive address	1F0h - 1F7h	3F6h - 3F7h
I/O - Secondary ATA drive address	170h - 177h	376h - 377h
I/O - Any 16-byte contiguous range	XXX0h - XXXFh	
Memory - Any aligned 2KB address range	Card must respond to locations XXX0h - XXXFh and X400h - X7FFh within the 2KB range	

In addition to mapping the registers, an interrupt request line must also be supported for I/O addressing. Normally IRQ 14 is used by ATA drives. When configured for memory-mapped registers, the socket interface does not define an interrupt line, therefore software polling must be used.

Supporting Two Drives

It is possible for two ATA drives to be simultaneously installed into PCMCIA sockets of the same HBA. When accessing these drives, some method must be used to individually select these drives as either drive 0 or drive 1. This is accomplished in a standard ATA environment via the daisy-chained cable with the cable-select signal or by jumpers (switches) on the drive. In the PCMCIA environment, a configuration register, called the Socket and Copy Register, can be used to identify two ATA PC cards mapped to the same address space. The copy number programmed into the Socket and Copy Registers is used by the HBA to differentiate drive 0 from drive 1.

Chapter 9

The Previous Chapter

The previous chapter discussed the PC Card ATA interface. A PC Card ATA interface provides a PC compatible hardware and programming interface that simplifies the job of implemented hard drive solutions in the PCMCIA environment. The chapter defined the various ways that a PC Card ATA can be mapped in the system along with the electrical interfaces that can be used. Differences between the PC compatible ATA implementation versus the PC Card ATA interface were also discussed.

This Chapter

This chapter focuses on the optional Auto-Indexing Mass Storage (AIMS) interface. The transfer mechanism is described, along with the registers that must be programmed to initiate the transfer.

The Next Chapter

Next, host bus adapter design is discussed, including the functionality that must be implemented, along with the optional features.

The AIMS Interface

The AIMS (Auto-Indexing Mass Storage) interface is designed to support cards that store large data structures to support functions such as imaging and multimedia. This interface creates a standard PC card interface for electronic cameras and other portable equipment requiring large amounts of data storage.

The interface uses a block transfer mechanism. Memory accesses occur through the card's registers. The specification describes the signal interface and the register set required by an AIMS card. The registers can be mapped

into either memory or I/O address space. See figure 9-1 for the AIMS interface pin definition.

Note that the AIMS interface specifies only an 8-bit data bus and seven address lines. The interface can be mapped into the host system's memory or I/O address space. The actual interface provided by the HBA will be a memory-only interface or the memory or I/O interface; however, the AIMS card only uses the pins illustrated in figure 9-1.

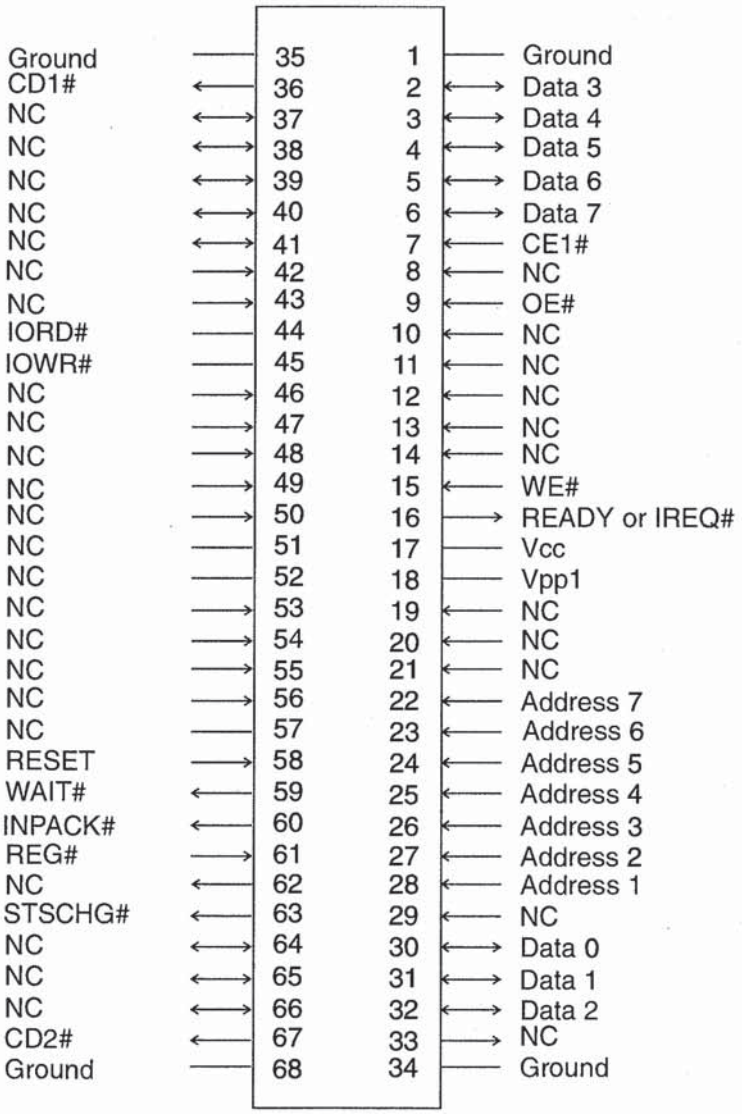


Figure 9-1. AIMS Socket Interface Signals

The AIMS Register Set

The memory array incorporated into an AIMS PC Card is accessed via a register set that includes:

- Address register (32-bit) — specifies the target memory address within the AIMS card that is being accessed.
- Data register (8-bits) — contains read or write data.
- Command register (8-bits) — contains the commands written by the AIMS device driver that specifies the operation to be performed.
- Mode register (8-bits) — provides status information regarding the card functions and controls the operation of the interrupts.
- Block count register (16-bits) — specifies the number of blocks to be erased when the erase command is issued to the command register.

Since AIMS cards can also support the memory or I/O interface, two additional registers are required:

- Configuration Option Register
- Pin Replacement Register

These registers are located in attribute memory address space at the location specified by the Configuration tuple within the CIS.

The Block Transfer

Transfers to or from an AIMS PC Card is controlled by a specific device driver that is aware of the AIMS transfer mechanism. This driver must program the AIMS card to perform the desired block transfer. The AIMS design is based on the expectation that flash memory will be employed as the memory array, and therefore, the block size is related to the flash block size for reading, writing, and erasing memory locations.

The following list describes the block transfer process initiated by the AIMS device driver:

1. Transfers are initiated by loading the address register with the starting block address to be transferred to or from the AIMS card. This requires four 8-bit writes to load the entire 32-bit address register.

2. Once the start address has been specified, then a command is issued to the command register. (Note that the mode register should be read to ensure the card is READY before writing to the command register.)

3. The AIMS card executes the command by either reading data from its memory array and placing it into the data register, or accepting write data into the data register and writing it to the memory array.

4. If no errors occur during the transfer, the address register auto-increments. The read or write transfer continues with the address being automatically incremented by the AIMS card each time another byte is successfully transferred.

5. The block transfer terminates when the device driver issues either an End of Read (EORD) or an End of Write (EOWR) command to the AIMS card.

6. When the AIMS card has stopped the transfer, sets any error conditions in the mode register and notifies the host system that the transfer has ended (by asserting READY for memory-mapped implementations or by asserting IREQ# for I/O-mapped implementations). The device driver then checks transfer status to determine the results of the transfer.

If a write operation is specified, the device driver would have issued an erase command prior to issuing the write command.

The AIMS Commands

A variety of commands are defined by the AIMS specification. Two basic types of commands are defined: type C and type D. Type C commands have a defined end state (e.g. an "end of write" command), whereas, the type D commands execute the specified command repetitively (e.g. a "write blocks" command). The AIMS commands are listed in table 9-1.

Table 9-1. Commands Supported by AIMS Cards

Name	Code	Type	Registers Used	Description
Write Blocks	38h	D	Address Command Status Data	Initiates block write operation. Address auto-increments until the write operation terminates when the "end of write" command is issued.
End of Write	4Fh	C	Command Status	Terminates a write block transfer. The AIMS card either generates an IREQ# or asserts READY upon completion of the transfer.
Read Blocks	40h	D	Address Command Status Data	Initiates block read operation. Address auto-increments until the read operation terminates when the "end of read" command is issued.
End of Read	3Fh	C	Command Status	Terminates a read block transfer. The AIMS card either generates an IREQ# or asserts READY upon completion of the transfer.
Erase Blocks	C0h	C	Address Block Count Command Status	Initiates a block erase operation. The operation continues until the specified number of blocks (contained the block count register) have been erased.
Execute Diagnostics	90h	C	Command Status	Initiates PC Card diagnostics. When the diagnostic completes the card reports any error conditions and notifies the system via READY or IREQ#.
Enter Diagnostic Mode	88h	C	Command Status	Places the card into the diagnostic mode. Note that the effects of accessing the AIMS registers is vendor specific when in diagnostic mode.
Return Internal Error Code	9Ah	C	Command Status Data	Provides access to error codes by reading the data register.
Write Verify Blocks (optional)	3Ch	D	Address Command Status Data	Initiates a write block transfer, like the write block command, except the card verifies the data just written prior to incrementing the address and accepting the next write data.

Accessing the AIMS Registers

The AIMS registers are identified by the PC Card address (A7:A1) and by indicating either a read or write transfer. These registers are mapped in either the host system's memory address space or I/O address space.

- When memory-mapped the registers are accessed with REG#=1 and the OE# and WE# commands are used to specify either read or write.
- When I/O-mapped the registers are accessed with REG#=0 and the IORD# and IOWR# command lines specify either a read or write operation.

Table 9-2 lists the registers and the address locations used to access each AIMS register. Note that the address offset applies to both memory- and I/O-mappings.

Table 9-2. AIMS Registers

Register Name	Read/Write	Address Offset
Address Register 0	Read/Write	00h
Address Register 1	Read/Write	02h
Address Register 2	Read/Write	04h
Address Register 3	Read/Write	06h
Block Count Register (low byte)	Write	08h
Block Count Register (high byte)	Write	0Ah
Command Register	Write	0Ch
Mode Register	Read/Write	0Eh
Data Register	Read/Write	10h
Vendor Unique Register	undefined	12h
Vendor Unique Register	undefined	14h
Vendor Unique Register	undefined	16h
Reserved	undefined	18h-1Eh

Chapter 10

The Previous Chapter

This chapter focuses on the optional Auto-Indexing Mass Storage (AIMS) interface. The transfer mechanism is described, along with the registers that must be programmed to initiate the transfer.

This Chapter

This chapter discusses the role of the PCMCIA Host Bus Adapter. Individual Host Bus Adapter functions are discussed. A functional block diagram of an HBA adapter is provided along with detailed explanations of each function.

The Next Chapter

The next chapter discusses the CIS and its role in the PC Card configuration process. The basic structure of the CIS configuration table required by I/O cards is described, along with the method used by configuration software to interpret the configuration table entries.

Introduction

As illustrated in figure 10-1, the PCMCIA Host Bus Adapter (HBA) resides physically between the PCMCIA host bus (usually an expansion bus such as ISA, EISA, Micro Channel or PCI) and the PCMCIA sockets. Since PCMCIA is host bus independent, this chapter focuses on the specific HBA requirements without detailing the exact nature of the expansion bus interface. Refer to the following MindShare books for information on other expansion buses: *ISA System Architecture*; *EISA System Architecture*; and *PCI System Architecture*, all published by Addison-Wesley.

Figure 10-1 also illustrates the software flow to and from the HBA. Two distinct software paths are illustrated:

- configuration and event notification path (PCMCIA specific)
- run-time path (normal program flow and execution)

Note that client drivers are shown in both software paths. These client drivers typically contain both PCMCIA specific code and standard run-time code (i.e. during execution of application program). The PCMCIA code within a client driver interacts directly with card services. The client driver is responsible for configuring the PC Card and programming the HBA so it can bridge the appropriate transactions from the host expansion bus to the PCMCIA sockets and cards. The client drivers are also responsible for processing PC Card status change events.

As illustrated in figure 10-1, client drivers do not access the HBA directly, but rather call card services, which in turn call socket services which access the HBA directly. Socket services provide a standard calling interface for card services to use. In this way, card services can call socket services functions to request that specific HBA functions be programmed as specified by the client driver.

Once the PC Cards have been configured, the PCMCIA-specific software has done its initial job and becomes dormant in memory. Application programs can now access the PC Card just as they do any other device that resides on the host's expansion bus. During PC Card configuration, the HBA is programmed to recognize addresses that reside on the PC Card. When the HBA recognizes an address belonging to a PC Card, it bridges the expansion bus cycle to the PC Card's socket.

PCMCIA software only runs again if the client driver needs to reprogram the HBA or if a PC Card status change event occurs. When the HBA detects a card event change it generates an interrupt that "wakes up" card services which calls socket services to determine the source and type of the status change event. Card services then calls the client driver, notifying it of the event. The client driver then processes the event as required.

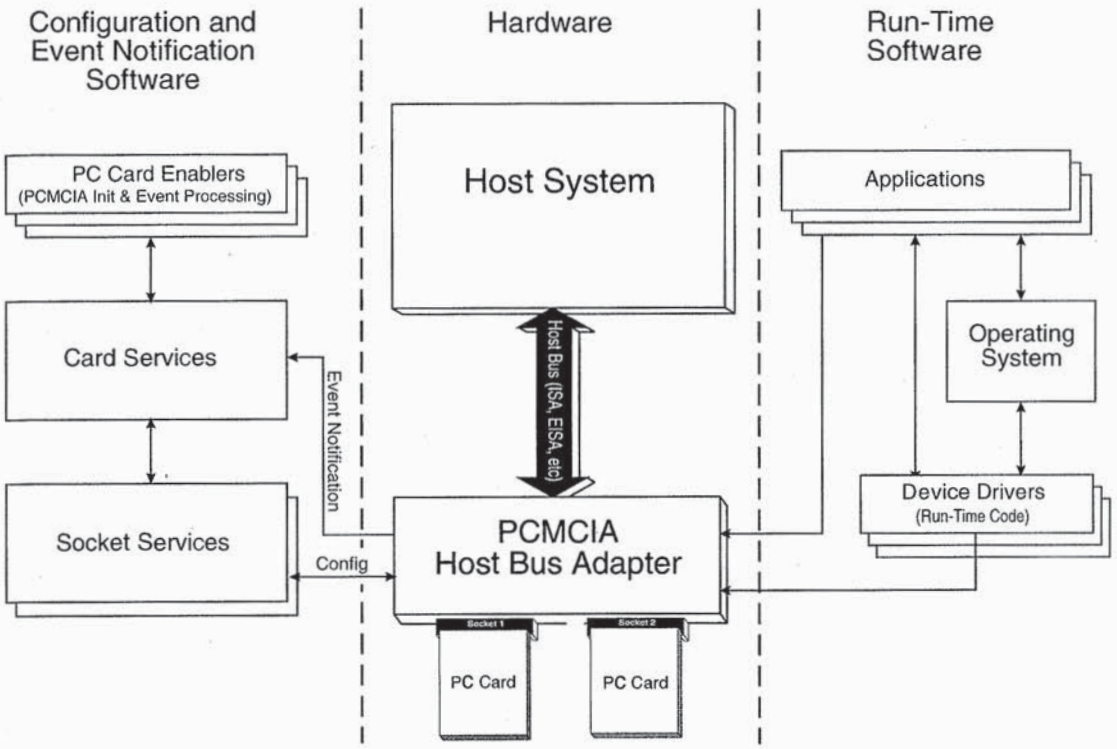


Figure 10-1. The PCMCIA Environment

In summary, the HBA is initially accessed by PCMCIA software that programs it to recognize accesses made to the PC Card. Once programmed, the HBA recognizes target addresses within the PC Card and passes the bus transaction to the card socket. The HBA also monitors status change events and generates an interrupt to inform card services of these events.

Host Bus Adapter Functions

The host bus adapter (HBA) provides the interface between the host bus and the PC Card sockets. This interface must be able to translate host bus accesses to PC Card socket accesses. Specific functions that must be supported by the HBA, include:

- Power switching
- Card detection
- Address translation

- Socket data buffering and control
- Socket data transfer timing and control
- Host bus transfer control
- PC Card interrupt steering
- DMA channel steering
- Socket status reporting
- Status change interrupt generation and steering
- Power Conservation (Power Management)

Note that a given HBA may support other interfaces including ATA (AT Attachment), AIMS (Auto-Indexing Mass Storage) and other custom interfaces. This chapter focuses on the memory and I/O interfaces and their related functions.

Figure 10-2 illustrates the primary functions associated with an HBA designed with two socket interfaces. HBAs are typically designed using an HBA chip supplied by one of several vendors. The functions in figure 10-2 incorporated into a specific chip can vary from vendor to vendor.

The Socket Interface

Since the PCMCIA socket interfaces are not bussed together, but rather are specified as independent socket interfaces, separate signal lines are required for each socket. This means that 68 pins must be included for each socket supported by an HBA. As a result, typical HBA implementations contain either one or two socket interfaces due to the number of pins required for the socket interface. (Some HBA designs may share some of the interface pins between sockets, reducing the total number of pins required for each socket.)

Maximum Number of HBAs

Some system implementations may require that four or more sockets be implemented. This means that more than one physical HBA chip is likely to be implemented (based on typical chip designs — two sockets/chip). Similarly, an additional HBA may be added in systems that have expansion slots. The theoretical maximum number of HBAs that a single system can support depends on the socket services software interface. With the Intel x86 binding (software function calling protocol) the maximum number is 256 adapters.

Other factors may also limit the number of HBAs that can be supported, such as the amount of space available in ROM or main DRAM.

Maximum Number of Socket Per HBA

The maximum number of sockets per HBA may also be determined by the socket services interface. The maximum number of sockets possible with the Intel x86 binding is 16 per adapter, governed by field width of the socket selection parameter within the Inquire, Get, and SetWindow functions.

Data Buffers / Transceivers

Each socket must be isolated from the expansion bus so that PC Cards can be inserted and removed when system power is applied. These buffers (usually transceivers) prevent transients from being introduced on the expansion bus signal lines when a PC Card is inserted or removed from the socket. Some HBA chips integrate the isolation buffers while others chips require external buffers.

Card Detection

The Card Detect pins (CD1# and CD2#) provide a way for the adapter to detect the presence of PC Cards when they are inserted into a PCMCIA socket. When power is applied to the system, the HBA can check the Card Detect pins to determine if power should be applied to the socket. The CD pins also provide notification that a PC Card has been either inserted or removed from a socket.

The HBA ties the CD1# and CD2# pins to Vcc through a 10K (or larger) pull-up resistor. Since these pins are tied to ground inside of each PC Card, when the card is fully inserted both pins will be pulled low, indicating that a card is installed in the socket.

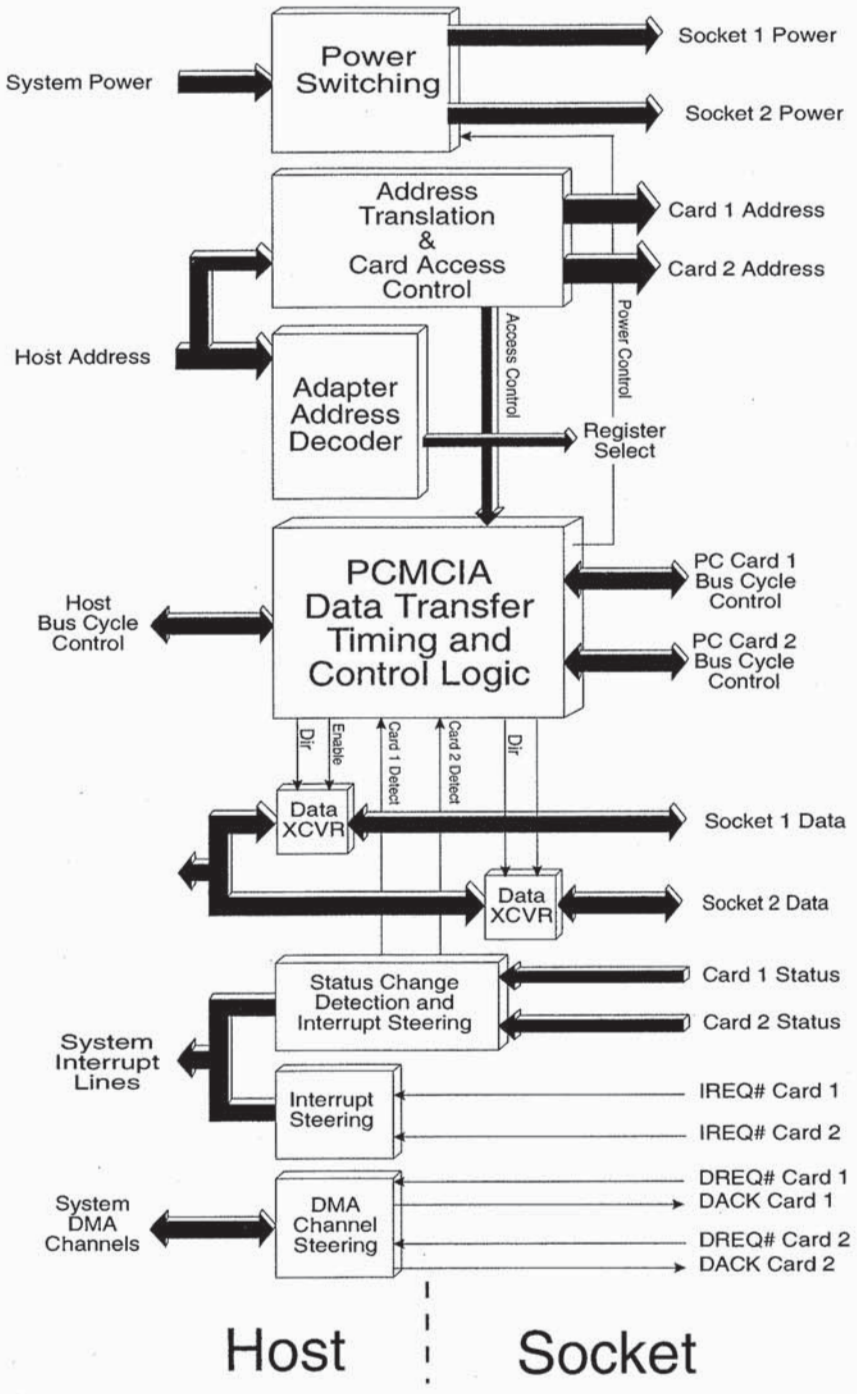


Figure 10-2. Host Bus Adapter Functional Block with Two Sockets

The HBA should be designed to generate a "socket status change" interrupt when a change is detected at the CD1# and CD2# pins. This interrupt, sometimes called the HBA, or management interrupt, notifies system software that some card event has taken place. Note that a status change interrupt can be caused by a number of card events (in addition to card insertion and removal). See the section entitled, "Card Event Notification" later in this chapter for information on other events that generate a status change interrupt, and for the steps taken by software when such an interrupt occurs. Ultimately, if card insertion or removal is detected, software will either configure the card or release resources that have been previously assigned to the PC Card.

Power Switching

The HBA must have the ability to switch different supply voltages to Vcc, Vpp1 and Vpp2 as required by the PC Card. The HBA must also be able to completely enable or disable Vcc, Vpp1 and Vpp2.

Vcc Power Controls

The initial Vcc that is applied to the PC Card socket depends on the version of specification that it supports. The following sections define the 2.1 HBA mechanism for applying Vcc followed by the new low voltage socket defined by the PC Card standard.

Vcc and 2.1 Compliant HBAs

The PCMCIA 2.x compliant HBA must be able to supply a Vcc of 5vdc and 3.3vdc to support dual-voltage cards. When a card is installed in release 2.x compliant systems, 5vdc is initially supplied to the card. Software then reads the CIS (the DEVICE_OC tuple specifies 3.3vdc support) to determine if the card has dual-voltage capability (operates at 3.3 vdc). If so, software must be able to direct the HBA to switch 3.3vdc to the card's Vcc pins.

In order to change Vcc, configuration software must remove power from the socket, select the new Vcc, and repower the socket. The PC Card will not retain any information from the previous power-up sequence. When Vcc is reapplied the entire initialization and configuration process must occur anew.

Vcc and Low Voltage Sockets

Systems based on the PC Card Standard must monitor the voltage sense pins (VS1# and VS2#) to determine the voltage that should initially be supplied to the Vcc pins (see table 10-1). The initial Vcc voltage applied depends on the voltages that the system and HBA is capable of delivering to the card's Vcc pin. Voltages supported are likely to include 3.3 vdc and 5 vdc. The X.X vdc referred to in table 10-1 is defined by PCMCIA as a low voltage supply to be defined in the future.

Table 10-1. Interpretation of Voltage Sense Lines.

Initial Power Required	Keying	VS1#	VS2#	Vcc at power up and CIS read
5 volts	standard*	1	1	5 volts applied if available, else no Vcc applied.
3.3 volts	low-voltage	0	1	3.3 volts applied if available, else no Vcc applied
3.3/5 volts	standard*			3.3 volts applied if available, else no Vcc applied.
X.X volts	low-voltage	1	0	X.X volts applied if available, else no Vcc applied.
X.X/3.3 volts	low-voltage	0	0	X.X volts applied if available, else 3.3 volts applied if available, else no Vcc applied.
X.X/3.3/5 volts	standard*			X.X volts applied if available, else 3.3 volts applied if available, else no Vcc applied.

* Standard keying refers to PC Cards keyed to fit into a 2.x compliant socket. These cards also fit into the low voltage sockets.

Vpp1 and Vpp2 Control

Socket pins Vpp1 and Vpp2 supply programming voltages for memory devices that require a special programming voltage. These pins can also be used to provide additional voltages for peripheral cards. Initially, these pins supply the same voltage as Vcc until the card's CIS is read to determine what special programming or peripheral voltages are required. PCMCIA recommends that Vpp1 and Vpp2 be tied to Vcc initially and then switched (as required by the card). PCMCIA also recommends that +12 vdc be available as an alternate supply for Vpp1 and Vpp2.

Some cards require the same supply voltage on Vpp1 and Vpp2, while others may require separate voltages be applied to Vpp1 and Vpp2. Whether the HBA can supply separate supplies to each Vpp pin is design dependent.

Address Translation

Translation of the address from the host bus to the PCMCIA socket may be required due to:

- The form of address used by the host expansion bus may need to be converted to the form expected by PC Cards. The nature of the translation depends on the nature of the host bus address.
- The maximum address space supported by the host system differs from the maximum address space supported by PC Card sockets.
- The PC Card responds to a fixed address range already allocated to another device in the system.

In short, the HBA must, if necessary, translate the host address to the form recognized by PC Cards, and remap the address presented on the host bus to the appropriate address within the PC Card.

Memory Address Mapping

The host address may need to be remapped to a different location within the PCMCIA address space for a variety of reasons including:

- The host and socket address space are not the same size.
- System software may constrain the addresses that a card can be mapped to within system address space.
- PC Card address decoder may not be programmable.

Direct Mapping

System addresses can be mapped directly to the same locations within the PC Card's memory address space. In this case, no re-mapping is required by the HBA. For example, assume that a 10MB flash memory card is installed in a host system whose expansion bus supports 16MB of address space. The 10MB flash memory card could be directly mapped within the system's address

space (as shown in figure 10-3). This example also assumes that the flash card's address decoder can be programmed to the range of addresses specified. If the flash card's address decoder can not be programmed, then the range of addresses that it responds to is fixed and the same fixed address range within system memory address space would have to be allocated in order for the PC Card to direct-map the card.

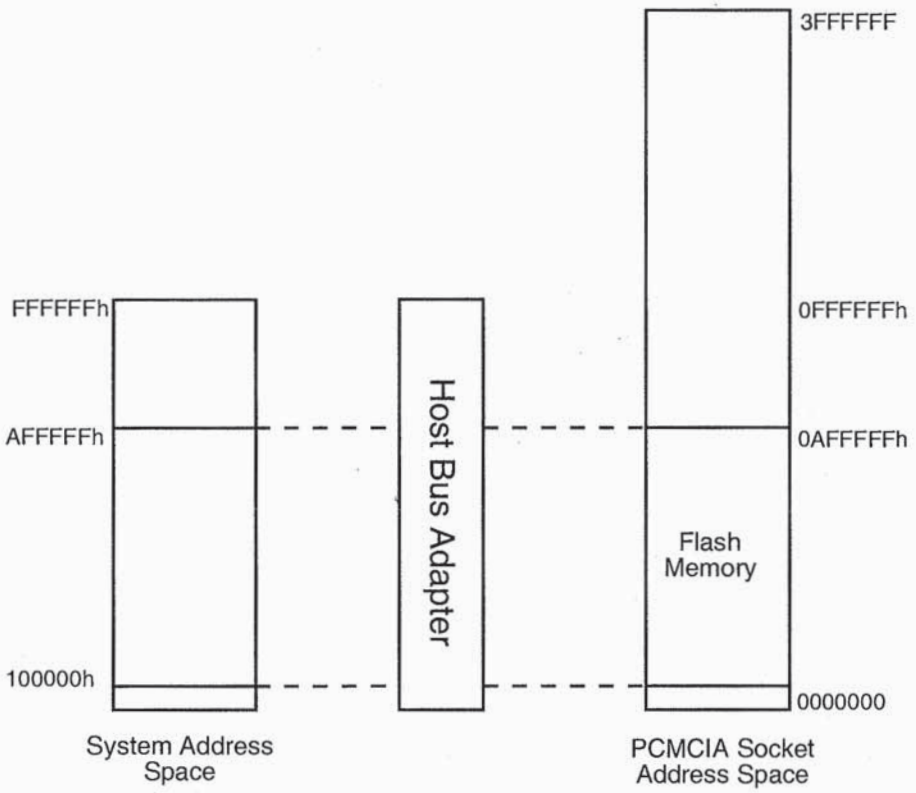


Figure 10-3. PC Card with Memory That Can Be Direct Mapped into the System Address Space.

Remapping the Host Address to PC Cards with Fixed Addresses

The previous example illustrates mapping a flash card with a programmable address decoder directly into the system address space. Next, consider a 10MB flash card whose address decoder is not programmable and whose ad-

dresses reside in the PC Card's common memory from address 0 to 10MB. In order to directly map the flash card into system memory address space, it must be mapped within the same system address space (0-10MB). Since much of the first megabyte of memory address space is reserved for other system devices and functions, this creates a problem in PC environments. Mapping the flash card in this space would cause data bus contention if an address within the range assigned to two memory devices was accessed.

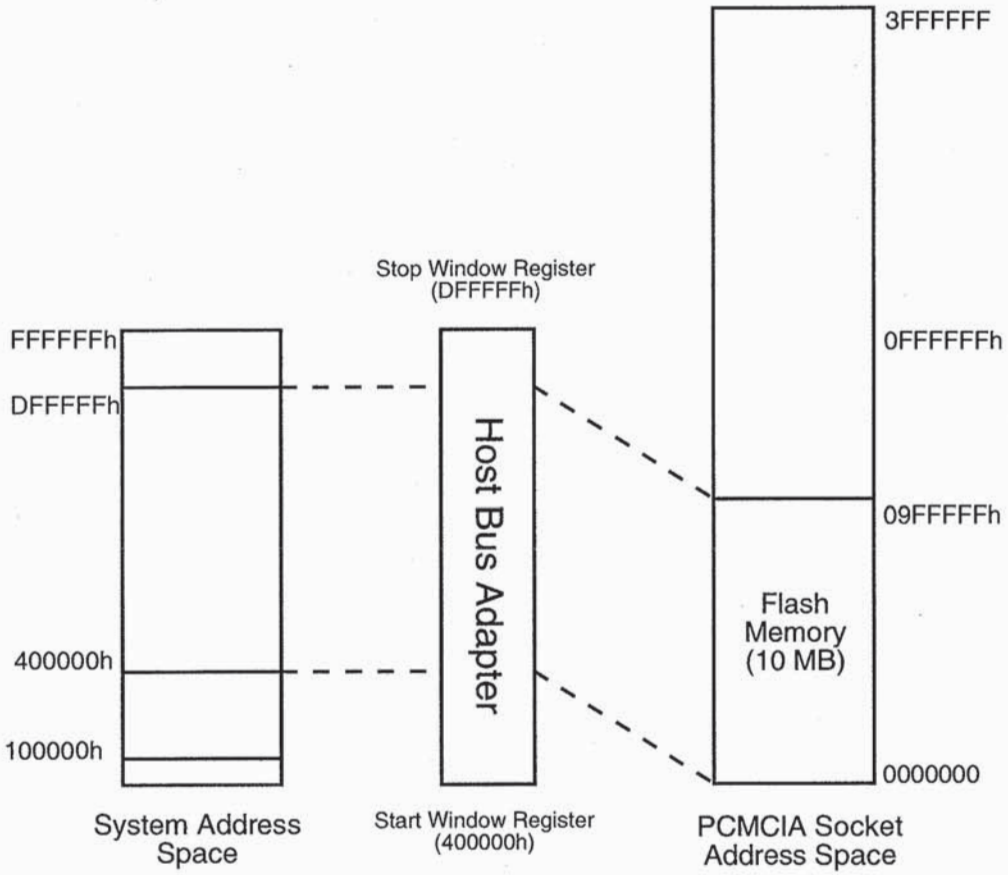


Figure 10-4. Example of Address Translation Logic Remapping the System Address to the Bottom of the Common Memory Address Space.

Successfully mapping the 10MB flash card requires that the address be mapped within system address space to a location that doesn't conflict with standard memory devices installed in the system. For example, consider a system containing system DRAM memory that is mapped up to 4MB. Refer to figure 10-4. The flash card could be mapped within system memory starting at

address location 400000h to DFFFFh. Since the flash card resides in common memory from address location 000000h to 9FFFFFh, the system address must be remapped to this address space in order to access the flash card. Consequently, the HBA must provide a means of remapping system addresses to the appropriate address space in the PC Card's address space.

System Address Space Smaller Than Socket Address Space

If the host system is incapable of addressing the entire 64MB of PC Card address space, then the HBA must have the ability to remap system address space such that all PC Card locations are accessible. Assume for example, that a 20MB flash card is installed in a socket whose HBA connects to an ISA host bus. Since the maximum memory address space supported by the ISA bus is 16MB, the system cannot possibly address all the locations within the 20MB flash card in a direct fashion. Therefore, remapping must be employed to permit access to all flash memory address locations.

This concept is shown in figure 10-5. To accomplish this remapping, system address space must be mapped twice: once to access the first 10MB address range and once to access the second 10MB address range. Using this technique, the HBA can be dynamically programmed to redirect, or remap accesses within the system address space (via an offset value) to different regions within the PC Card's memory array, permitting access to the entire 64 MB of address space.

Note that the technique described above is common in 8088-based systems that employ a maximum of 1 MB of memory address space, much of which is not available for mapping PC Cards. The host address space typically used by PC memory cards is within the address range from D0000h to DFFFFh.

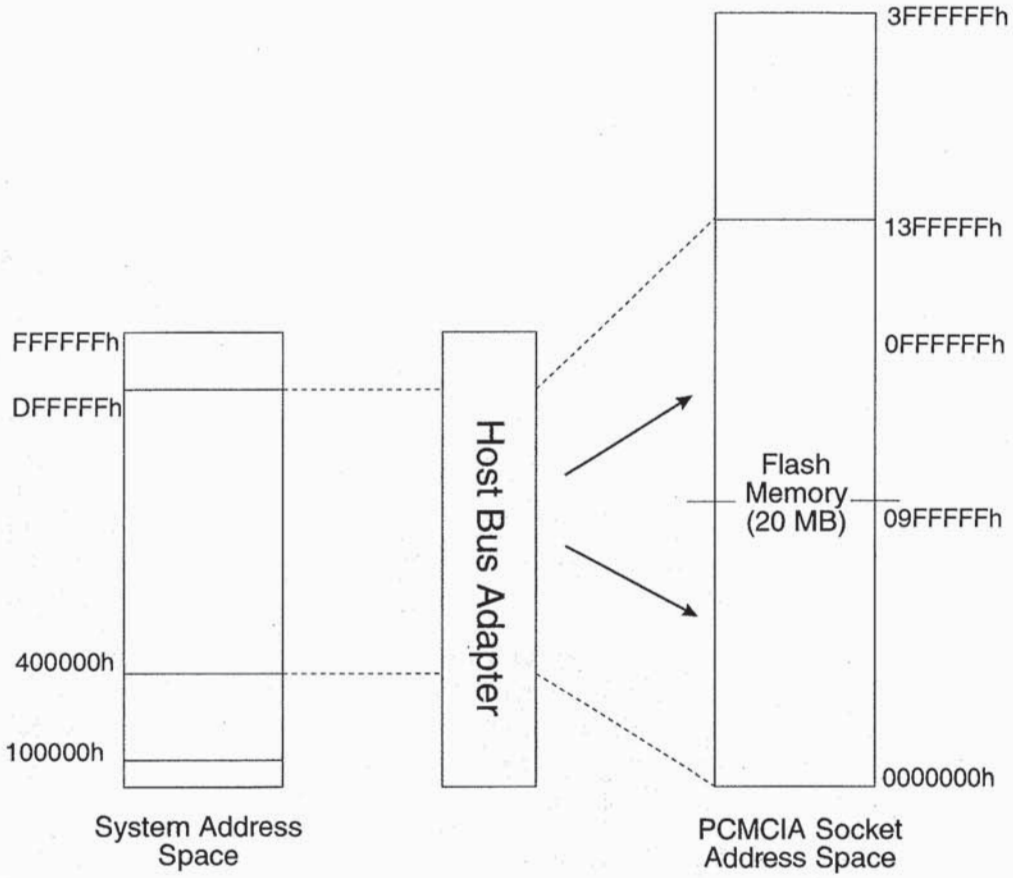


Figure 10-5. Example of Small System Address Range Being Remapped to a Larger PCMCIA Memory Device.

System Address Space Larger Than Socket Address Space

Accesses can also be made to PC Cards from system addresses beyond the 64MB maximum address range supported by the PC Card socket. Once again the system addresses are remapped to locations within the PC Card's address space. Refer to figure 10-6.

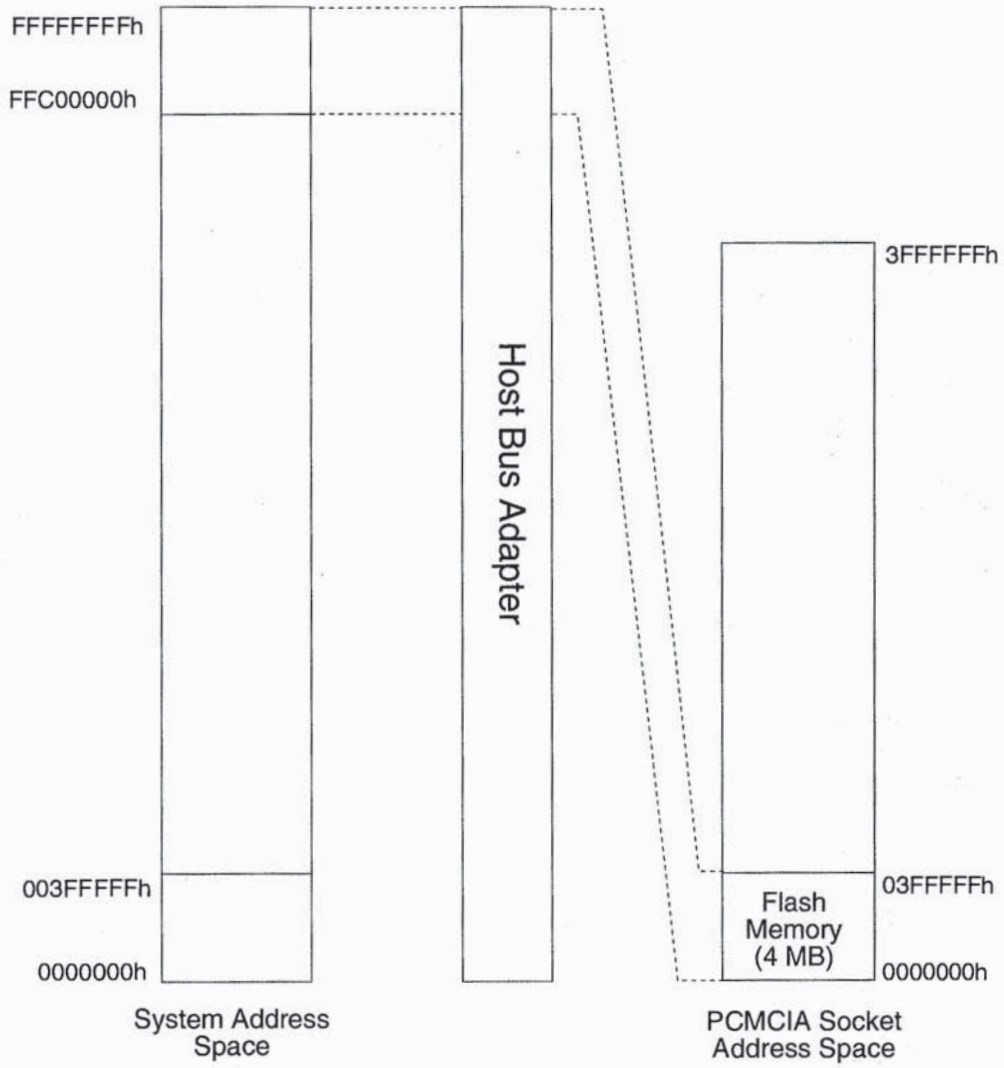


Figure 10-6. Example of System Address Exceeding PCMCIA Address Range.

Memory Address Windows

Address windows are used by HBAs to determine the range of host system memory addresses to which a particular card responds and to remap the system memory address to a location within the PC Card's memory address space. The HBA address windows are typically implemented with programmable window address registers, permitting software to program the range of memory addresses to which the PC Card in a given socket can respond, and

what PC Card memory address it should be remapped to. Note that a given PC Card may have several different blocks of addresses that it must respond to, each requiring a separate address window. HBAs typically have several address windows that can be programmed for both memory and I/O devices.

Each address window consists of a:

- Start window address register
- Stop window address register
- Window offset address register (mapping register)

When the host system accesses some device within the system, the HBA must determine if the address on the host bus is within any of the programmed address windows. If so, the host access is to a location within the PC Card installed in a socket, and the HBA initiates a transfer to or from the socket. Otherwise, the host access is ignored. In short, the HBA acts as the PC Card's address decoder by passing only those transactions to the PC Card that are intended for it.

In addition to determining if the host address targets a socket, the HBA also remaps the address if necessary. Remapping is typically accomplished via an offset register. This register is programmed with a value that, when added to the host address, redirects (remaps) it to the desired location within the PC Card's memory address space. Refer to figure 10-7.

When the host address must be mapped to a lower address, the offset value equals the number of address locations that must be subtracted from the host address. Since the HBA always adds the contents of the offset register to the host address, the programmer uses the two's complement of the offset value, resulting in a negative offset (as shown in figure 10-7).

Overlapping Memory Windows

When two PC Cards are installed, each is mapped into it's own socket address space. Normally, each will be accessed within separate host address ranges. This prevents contention between the two devices. If, however, the host has insufficient address space to map both devices into separate ranges, then the address windows of the two sockets may overlap, creating the potential for contention (as shown in figure 10-8).

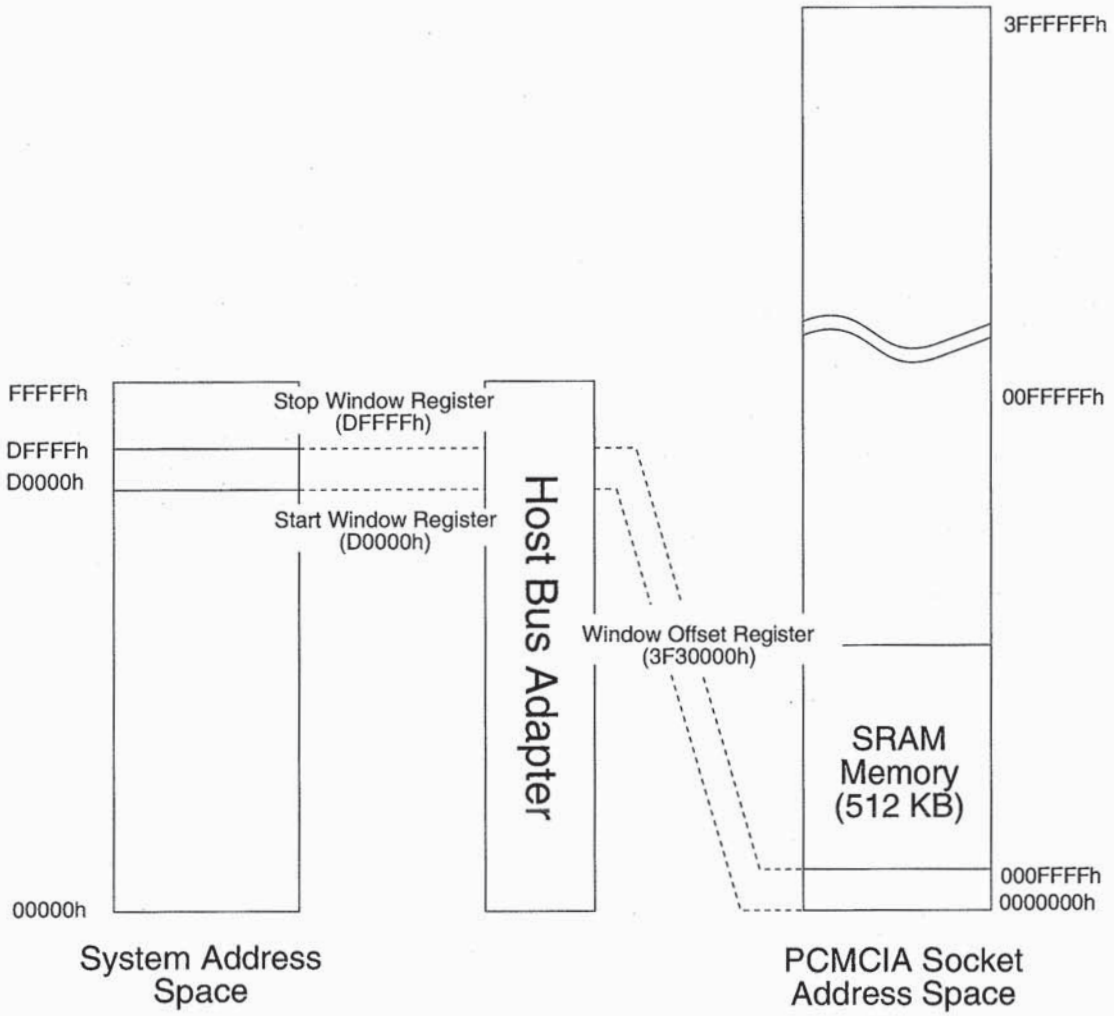


Figure 10-7. Registers Define the Size of the Memory Window and the Size of the Offset for Remapping the System Address.

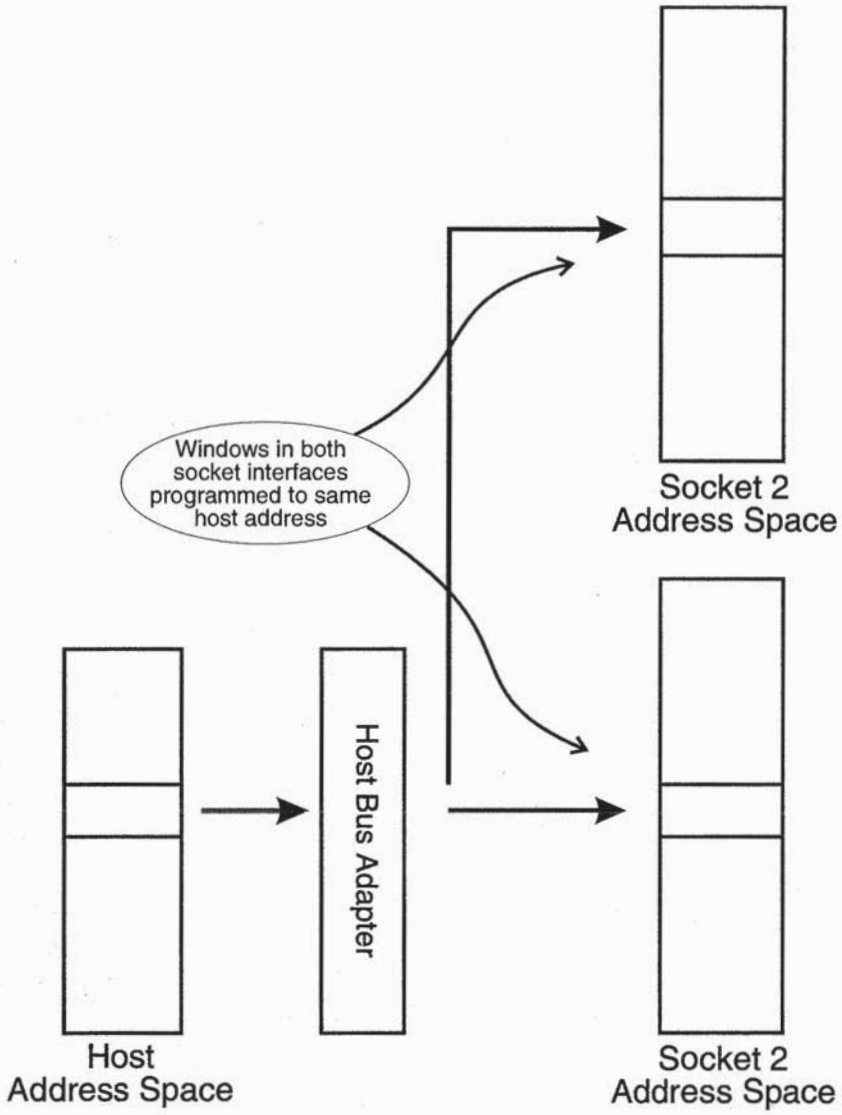


Figure 10-8. Example of Overlapping Memory Windows Causing Contention

When memory windows overlap, only one window can be enabled at a time, thereby preventing contention. This highlights the requirement that memory windows, once programmed, support the ability to be enabled and disabled under software control.

I/O Address Mapping

PC Cards containing I/O registers must be mapped into the system's I/O address space. If the system has only memory address space, then the HBA must map memory addresses into the PC Card's I/O address space. The following discussion assumes that the system has I/O addresses.

Direct Mapped I/O Addresses

Figure 10-9 illustrates PC Card registers being mapped directly to the corresponding system I/O addresses. The HBA in this example, would bridge the expansion bus cycle to the PC Card when an I/O address is detected within the ranges of the addresses used by the PC Card. In this example, two I/O windows must be used to specify the address ranges required by the card.

Overlapping I/O Windows

Sometimes I/O windows must be programmed to overlap with addresses of I/O devices residing elsewhere in the system. Consider the example in figure 10-10. Assume that the HBA has two I/O address windows for a given socket, and that the PC Card requires three I/O address ranges that are spread widely across the system I/O address space. Since only two address registers exist, one must be programmed to encompass two of the three address ranges. This large window may encompass address locations used by another PC Card in a different socket or locations used by some other device residing elsewhere in the host system.

When an I/O read access occurs from a location within the large address window, the HBA responds by starting a data read transfer from the socket. The PC Card decodes the socket address to determine if the address is to one of its registers. If so, the INPACK# (Input Acknowledge Port) signal is asserted by the PC Card and the transfer completes from the PC Card. The INPACK# signal enables the data transceiver so that data from the socket is returned over the host data bus.

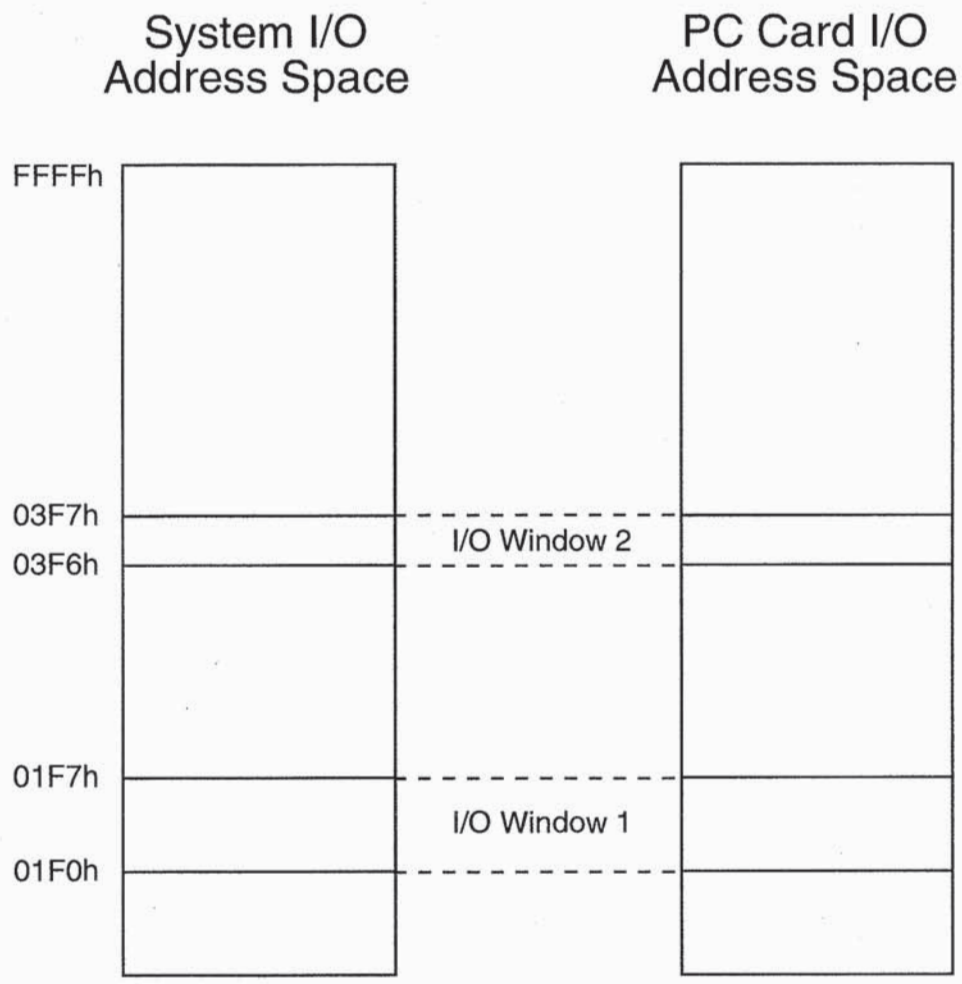


Figure 10-9. PC Card I/O Addresses Mapped Directly to System I/O Addresses

If the address decoded by the PC Card is not recognized, the INPACK# signal is not asserted. Since INPACK# is not asserted, the HBA also ignores the host access and the socket's data bus transceivers remain disabled.

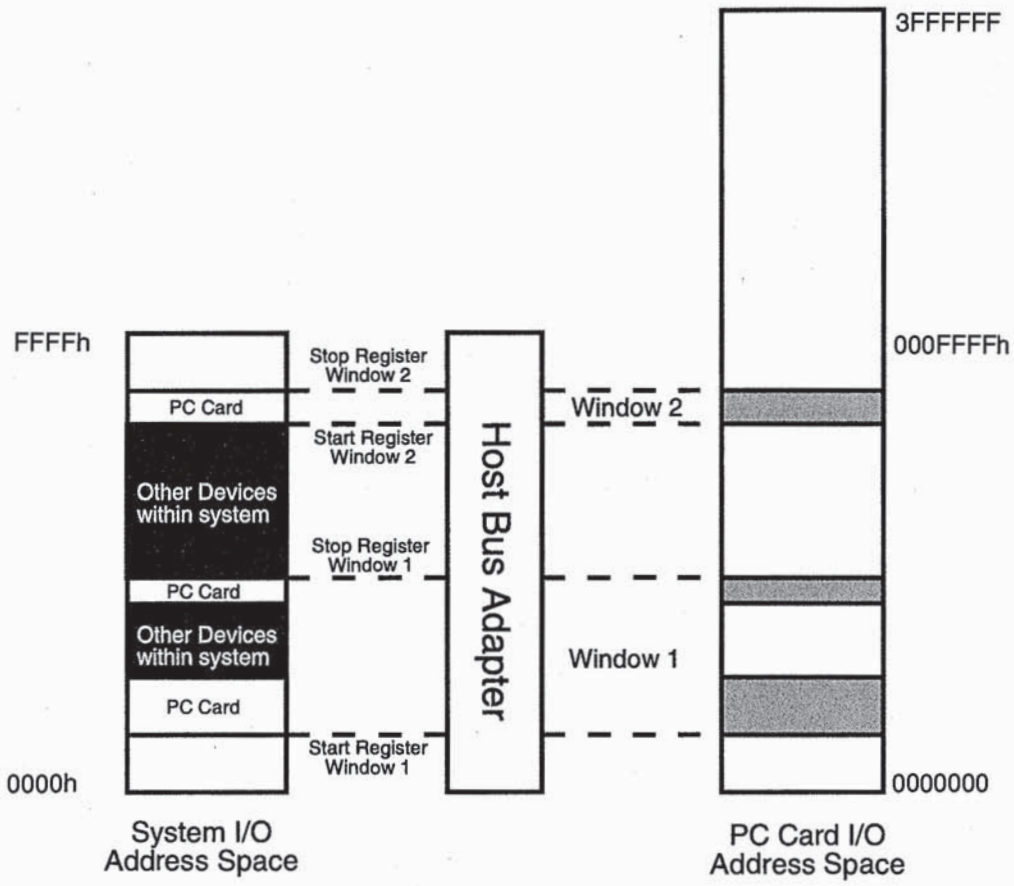


Figure 10-10. Example of I/O Window Overlapping Addresses of Other I/O Devices in the System.

It may be necessary to program I/O address windows that overlap between two sockets (as illustrated in figure 10-10). When I/O windows overlap, PCMCIA permits simultaneous reads from both sockets, relying on the PC Card decode and the INPACK# signal to prevent contention.

Other Information Associated with Address Windows

When the HBA detects that a host address is within the range of an address window, it initiates a data transfer to or from the PC Card. Since PC Cards can contain devices of varying speeds, it is also necessary to program the access time of the device residing within the targeted address window.

Additionally, PC Card device size information must also be associated with I/O address windows. This is necessary since I/O registers within PC Cards can be either 8-bit or 16-bit; therefore, the data path(s) used to transfer data to and from the PC Card can vary. The HBA must be able to detect I/O device size to determine which data path or paths the I/O device uses.

When client drivers read the CIS, they determine PC device speed and size and program this information into the HBA window(s) allocated for the card. In this way, the HBA knows the exact nature of the transfer it must run to the device being accessed. (For more information refer to the section entitled "PC Card Device Size" later in this chapter.)

Socket Transfer Timing and Control

The socket transfer timing and control logic has multiple interfaces: a host bus interface and one or more socket interfaces. When a read or write transfer is detected on the host bus, the address translation logic determines if the address is intended for one of the PC Cards, while the timing and control logic determines whether the access is a read or write to memory or I/O. When an address falls within one of the address windows, the address translation logic notifies the timing and control logic, triggering the appropriate socket access.

Access must be made to the socket to either read or write data based on the type of host cycle being run.

Interface Control

The HBA configures each socket at power up as a memory-only interface. However, once the card's client driver determines its configuration requirements, an alternative socket interface may be required. For example, the client driver must reconfigure the HBA from a memory-only to a memory or I/O interface when it detects an I/O card. Since some signal definitions change when a socket is configured for memory or I/O, the HBA must multiplex selected lines to alternative functions. A given HBA may also support other interface types such as DMA or AIMS, requiring additional socket interface redefinition.

Socket Access Timing

PCMCIA specifies standard cycle timing for common memory, attribute memory and I/O registers. (Refer to the Chapters entitled, "The Memory Interface" and "The Memory or I/O Interface" for details regarding cycle timing for the respective card types.) The timing and control logic accesses memory cards based on the timing specified in the CIS. The cycle timing of a PC Card's memory is specified within its CIS during card initialization. Each bank of memory with different cycle timing must be listed separately within the CIS. Each memory bank can be accessed via a separate address window that specifies the cycle timing required by the target bank. Cycle timing information is loaded into a card's memory or I/O window registers during card configuration. The PC Card standard requires that access timing information be programmed as part of the address window definition for each socket/device.

Stretching Socket Access Timing

PCMCIA also incorporates a WAIT# signal that is used for stretching the access timing. This capability is optional but typically supported for devices that may not always be able to respond to a socket read or write within the programmed timing.

Word or Byte Access

Host systems may have the ability to access single byte locations (from either even or odd locations) or entire words in a single bus cycle. When the HBA detects that an address being accessed by the host system resides in a given socket, it translates the address, if necessary, and starts a socket transfer. The address specifies the start location being accessed within the PC Card, but does not specify whether a single byte is being accessed or an entire word. The HBA must be able to detect the size of the host transaction and assert CE1# and/or CE2#, commanding the PC Card to transfer one or two bytes of data. For example, an ISA system uses SA0 and SBHE# to indicate the number of bytes being transferred. The HBA must translate these signals to the form understood by the PC Card (A0, CE1# and CE2#).

The actual combinations of CE1# and CE2# asserted by the HBA depends on the size of the host bus (as shown in table 10-2).

Table 10-2. Address Sent to Socket

Host Transfer Size	Location(s) Accessed	Socket Signals Asserted A0	CE1#	CE2#
8-Bit	Even Byte	0	0	1
8-Bit	Odd Byte	1	0	1
16-Bit	Even Byte	0	0	1
16-Bit	Odd Byte	1	1	0
16-Bit	Word	0	0	0

PC Card I/O Device Size (IOIS16#)

PC Card memory devices are always implemented as 16-bit devices. However, I/O PC Card can implement either 8-bit registers, 16-bit registers, or both. Device size is read from the card's CIS and programmed into the I/O address window, or alternatively, the IOIS16# signal can be sampled by the HBA to distinguish between accesses to 8-bit or 16-bit registers.

Some host expansion buses include device size lines that are asserted by the addressed target, informing the host system of the size of the device being accessed. The data path over which the data is expected and the bus cycle timing varies depending on the device size reported. For this reason, the size information reported by a PC Card may need to be transferred to the host system so that transfer timing and data path steering can be adjusted by the host system (e.g. the ISA bus controller) A given HBA will be designed to control both the socket access and the host access, ensuring that the transfer appears correctly to both.

Card Interrupt Steering and Handling

During initialization, a system interrupt request line is assigned to each PC Card that uses interrupts. The HBA is programmed to steer the PC Card interrupt (IREQ#) to the system interrupt request line allocated to the card

during the configuration process. Figure 10-11 illustrates ISA interrupt steering for a dual socket HBA. The number of ISA interrupts that an HBA supports is implementation specific.

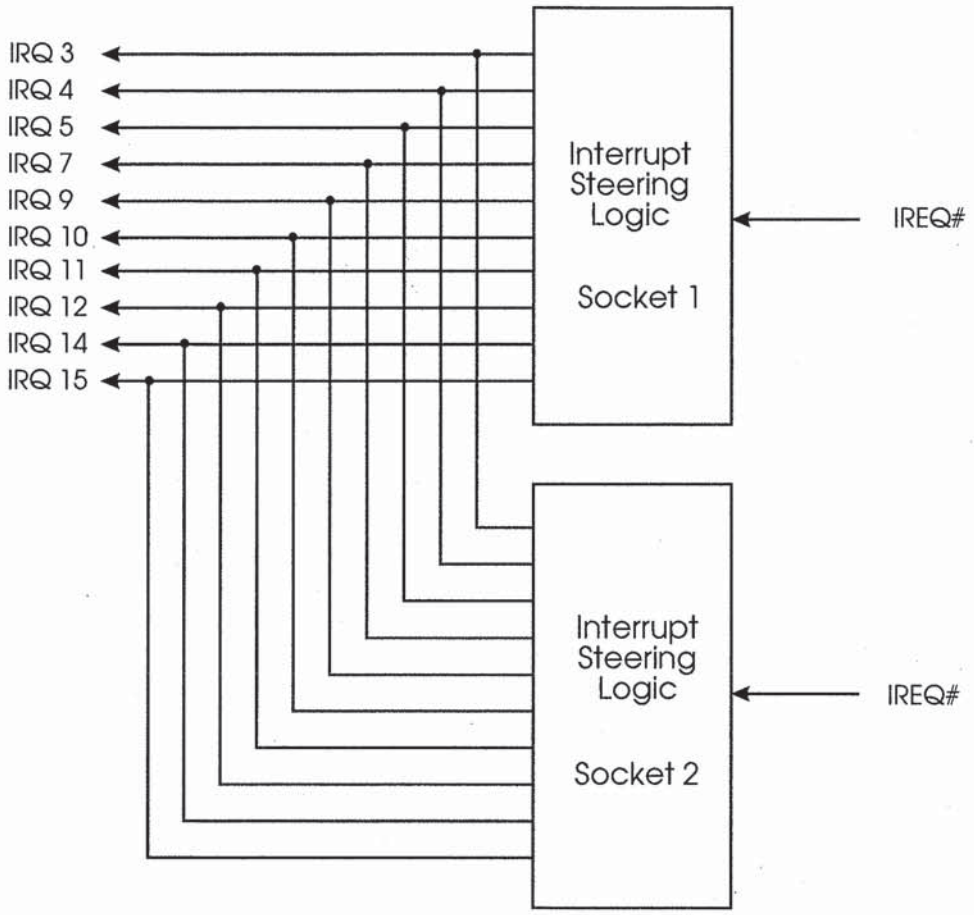


Figure 10-11. HBA Interrupt Steering in an ISA System

The IREQ# signal sent from a PC Card may be signaled by one of the following methods:

- Level mode
- Pulse mode

PC Cards must be designed to support level mode interrupts, whereas, pulse mode interrupts are optional.

Level Mode Interrupts

Normally, a PC Card is programmed for level interrupt mode. When in level mode, the IREQ# pin is pulled up to Vcc on the PC Card and asserted low to signal an interrupt. The interrupt is kept asserted until the interrupt service routine reads the PC Card's status register, thereby resetting the interrupt indication and causing IREQ# to be deasserted.

As illustrated in figure 10-12, when level mode interrupts are used the HBA must invert the IREQ# signal from the PC Card to generate a positive-edge triggered interrupt required by ISA systems. While the IREQ# signal is asserted by the PC Card using level mode, no other device can trigger a positive-edge interrupt (the interrupt line is driven high by the HBA from the time of trigger until the interrupt is serviced). Similarly, when no interrupt is being asserted by the PC Card the HBA continually drives the IRQn line low also preventing other ISA devices from generating a positive-edge trigger. Another device attempting to trigger a positive-edge will be unsuccessful and, more importantly, hardware damage can result.

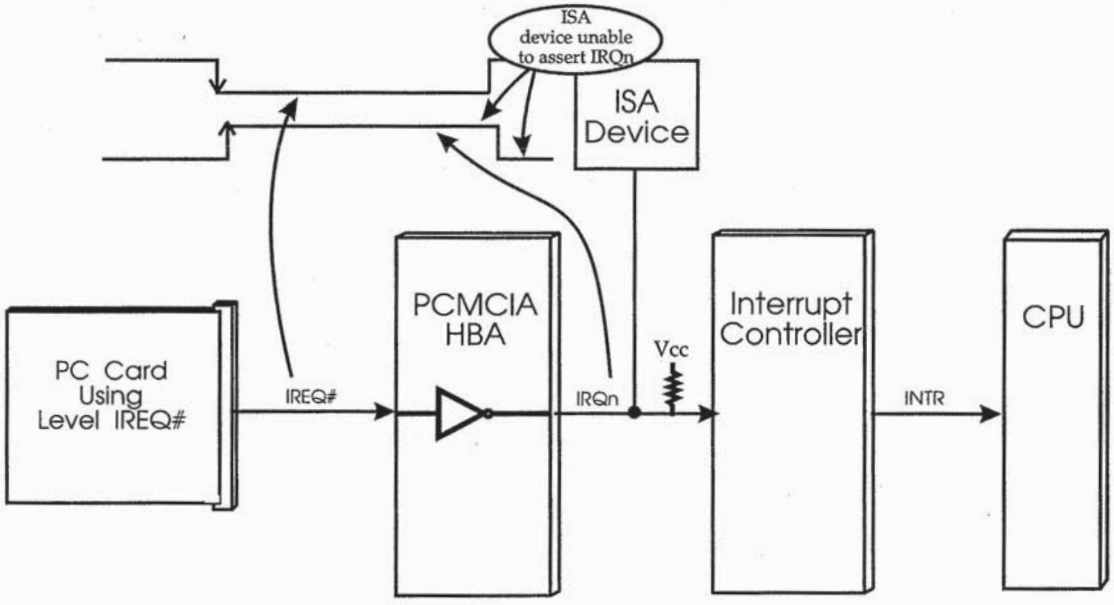


Figure 10-12. ISA Interrupt Sharing Not Permitted with Level Mode IREQ#

Level mode interrupt signaling can be used in ISA systems, where no sharing is supported, or in host environments permitting interrupt sharing via low level interrupt triggering, such as PCI and EISA systems.

Pulse Mode Interrupts

Since the level interrupt mode defeats the capability of interrupt sharing in an ISA system, pulse mode interrupts were implemented to allow sharing. Pulse mode interrupts allow a PC Card to share the same IRQn line with another device. On the trailing-edge of the negative pulse, a positive edge trigger will be recognized by the ISA interrupt controller. Note that the pulse must be equal to or greater than 0.5 microseconds in width. The HBA must use an open collector driver to permit other devices to drive the same IRQn line.

When an interrupt is not being triggered by the PC Card the IRQn line is pulled to Vcc. This permits another ISA device to trigger an interrupt by driving the IRQn line low and then cease driving it, thereby, causing a negative pulse whose trailing edge creates the positive-edge trigger. ISA devices designed to share interrupts also use an open collector driver for their IRQn line.

Only during the negative transition of the pulse are other devices prevented from triggering an interrupt. As a result using pulse mode interrupts to permit interrupt sharing in ISA hosts may still result in conflicts when two simultaneous interrupts are triggered by the devices sharing the interrupt line. Additionally, two interrupt pulses that do not overlap but that occur in close proximity to each other may also cause an interrupt trigger to be missed.

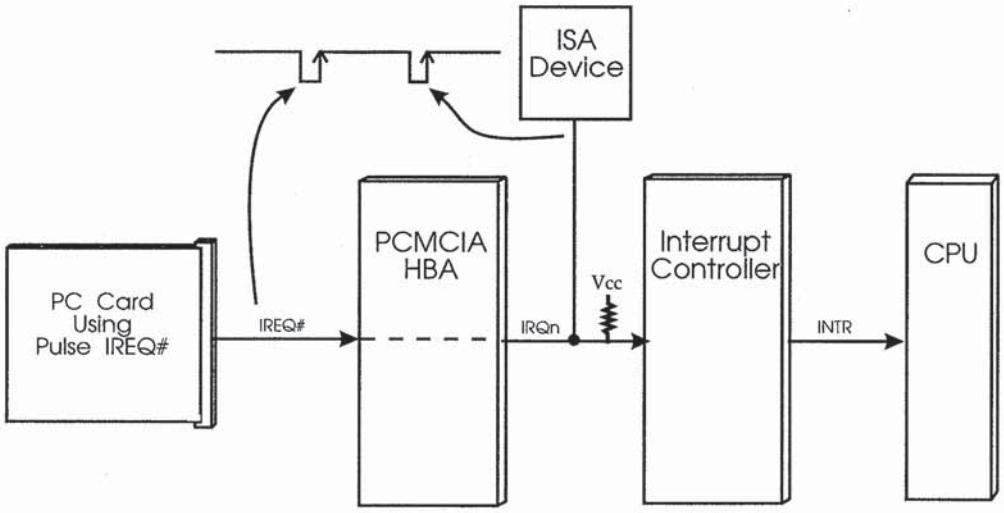


Figure 10-13. Pulse Mode Interrupts Permit Interrupt Sharing in an ISA System.

Card Event Notification (The Status Change Interrupt)

PC Cards can report PCMCIA status events such as low battery warnings, write-protect switch position change, READY indications, and card detection. When the socket signals a status change, the HBA must be able to generate an interrupt, notifying PCMCIA software (card and socket services) that a status change event has taken place. The PCMCIA status change interrupt (sometimes called management interrupt) must be steered to the appropriate system interrupt line under software control. The HBA should also have the ability to mask out specific status change events under software control so that they do not generate an interrupt.

The HBA must also implement status registers indicating which socket(s) experienced a status change event, which status event has occurred and the current state of each status indicator. During the interrupt service routine, socket services will read these status registers within the HBA to determine which socket or sockets have encountered a status change. When socket services has been notified that a status change has occurred at a given socket, it then reads another status register within the HBA to determine the actual event causing the interrupt.

DMA Support

DMA support is an optional feature for a PC Card compliant system. As illustrated in figure 10-14, HBAs supporting DMA transfers must contain logic to:

- select the PC Card's DREQ# signal from one of three pins: SPKR#, INPACK#, or IOIS16#.
- deliver the DACK signal from the ISA bus to the PC Card's REG# pin.
- deliver the TC signal from the ISA bus to the PC Card's OE# pin upon completion of a DMA write transfer.
- deliver the TC signal from the ISA bus to the PC Card's WE# pin upon completion of a DMA read transfer.
- steer DREQ# from the PC Card to any of the seven ISA DREQ lines.
- steer any of the seven DACK# signals from the ISA bus to the PC Card's DACK pin.

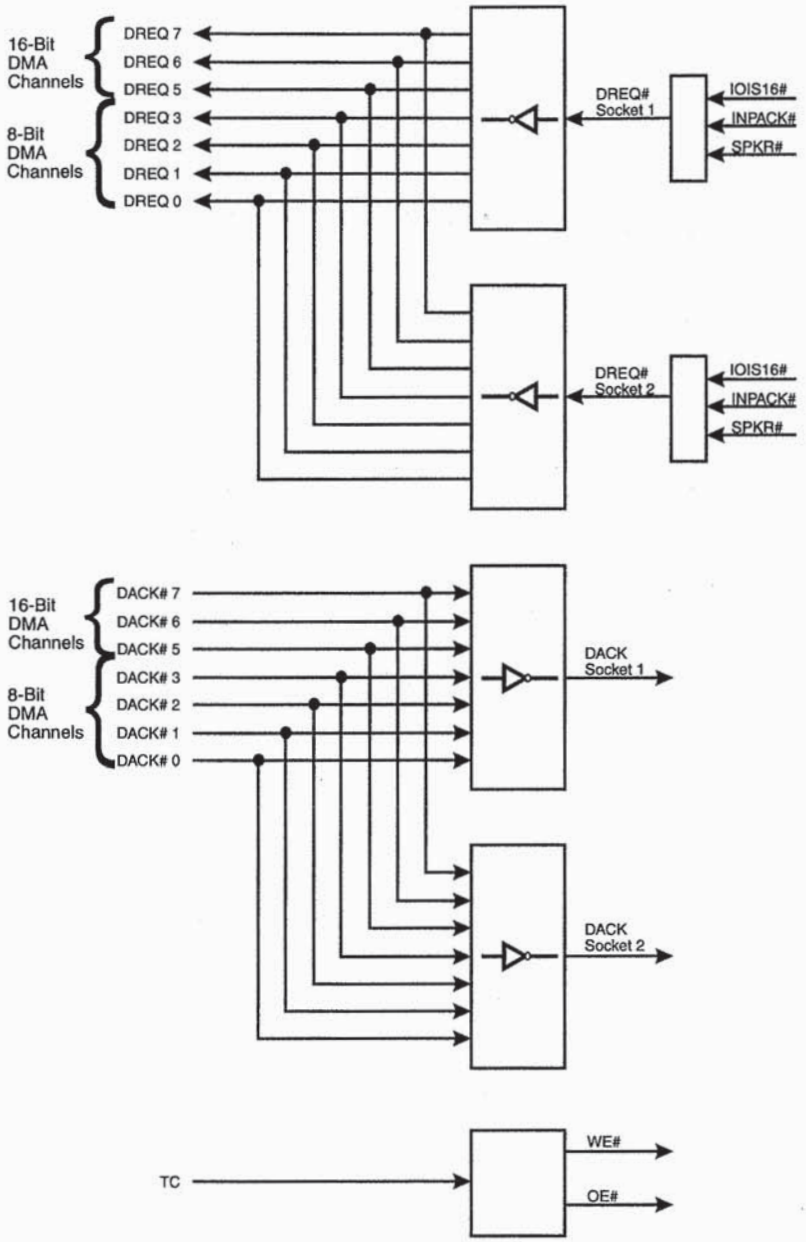


Figure 10-14. HBA Functions Required to Support PC Card DMA

The HBA also determines the width of the DMA transfer by monitoring the Expansion bus and asserting the card enable pins to reflect the transfer size (either 8- or 16-bits). Refer to the Chapter entitled "The DMA Interface" for additional information.

Power Conservation Modes

Most PCMCIA HBAs support some form of power conservation. For example, when no PC Card activity has occurred within a specified time period, as many functions as possible are powered down to conserve power. Only the logic necessary to detect card insertion and removal and system bus activity must remain active. Depending on the HBA design, it may also support additional power saving modes (where even more of the adapter functionality is powered off).

Card Lock Mechanism

PCMCIA includes optional support for card interlock mechanisms. A PC Card interlock has two major advantages over systems without an interlock.

1. Prevents the user from removing a PC Card while the card is in use. Such implementations may employ motor driven mechanisms used to insert and eject PC Cards. If the user requests that the card be ejected, software can remind the user that an applications currently using the card should be closed before removing the card.
2. Prevents PC Cards from being removed from a system by unauthorized personnel. Security mechanisms may employ a simple mechanical lock that prevents a card from being ejected from the socket. A key may be required to release the lock. These mechanisms may also be motor driven and require that a password be entered before software will start the motor to eject the card.

One major obstacle to the more complex solutions is the additional weight, size, and power required, making them less attractive in the mobile computing environment. The card lock functionality is optional and currently is not in widespread use.

Error Detection and Correction (EDC)

Support exists within socket services to support EDC, thereby providing a method to enhance reliability of HBA designs. The author is unaware of any current HBA solutions that implement error detection and correction generation. Since EDC generators are not typically used, the design and implementation of EDC generators is not covered in this book.

Part Three

PC Card Design

Chapter 11

The Previous Chapter

The previous chapter discussed the role of the PCMCIA Host Bus Adapter. Individual Host Bus Adapter functions are discussed.

This Chapter

This chapter discusses layer one of the metaformat, commonly referred to as the card information structure, or CIS. The chapter details the role of the CIS in the PC Card configuration process. Tuples are also introduced and their format and structure are described. The basic structure of the CIS's configuration table required by I/O cards is also described.

The Next Chapter

Configuration registers are discussed in the next chapter, providing a complete description of each register specified by the PC Card standard. Configuration register implementations for both single and multiple function cards are covered.

Overview

PC Cards include a data structure called the Card Information Structure (CIS) that is stored in non-volatile memory. The CIS provides a method for software to determine what kind of PC Card is installed, along with its speed, size, and the system resources required by the card. Having determined this information, the PCMCIA Host Bus Adapter (HBA) can be programmed to allow access to the PC Card, and the card itself can be configured by writing to its configuration registers. Configuration registers are required by I/O cards but are optional for memory cards. Both the CIS and configuration registers are mapped in the attribute memory space.