Navy Personnel Research and Development Center

San Diego, CA 92152-6800 TN-90-11 March 1990

A Portable Courseware Architecture

Brian Thomason
Brian Van de Wetering
Roger Booth

NPRDC-TN-90-11 March 1990

A Portable Courseware Architecture

Brian Thomason
Brian Van de Wetering
Roger Booth
Systems Engineering Associates
San Diego, California 92109

Reviewed and released by
Wallace H. Wulfeck, II
Director,Training Technology Department

Navy Personnel Research and Development Center
San Diego, California 92152-6800

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE March 1990	3. REPORT TYPE AND DATE COVERED Interim--Sep 88-Feb 90

4. TITLE AND SUBTITLE
A Portable Courseware Architecture

5. FUNDING NUMBERS

6. AUTHOR(S)
Brian Thomason, Brian Van de Wetering, and Roger Booth

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Navy Personnel Research and Development Center
San Diego, California 92152-6800

8. PERFORMING ORGANIZATION REPORT NUMBER
NPRDC-TN-90-11

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of the Assistant Secretary of Defense
(Force Management and Personnel) (Room 3E808, Pentagon)
Washington, DC 20301-0000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Many vendors produce high-performance, low-cost training hardware, but bundle their products with proprietary software interfaces. Because these interfaces are proprietary, courseware and authoring systems written to operate on one set of hardware will not run on a competitors's hardware. Expensive reprogramming is needed to adapt to new hardware. These reprogramming costs can be eliminated by adopting standard software interfaces. The objectives of this effort were to describe and develop a standard software interface that will allow training systems to be assembled from separate "plug-and-play" components in the same way that stereo systems can be assembled from separate speakers, amplifiers, and other components. The Portable Courseware (PORTCO) architecture consists of two interfaces, the Device Services Interface and the Device Handler Interface. It also contains three layers: application, routing and configuration, and device handler. This architecture should allow applications software to run on any compliant set of hardware components. The series of reports describing the PORTCO architecture should direct development of portable MS-DOS applications and standard peripheral device handlers. This report provides an overview of the PORTCO architecture and should be of interest to all who are concerned with computer-based training.

14. SUBJECT TERMS Courseware portability, computer-based training, interactive courseware, virtual device interface			15. NUMBER OF PAGES 26 16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UNLIMITED

Foreword

This document describes a draft specification for an architecture for interactive courseware delivery systems. It was developed under the sponsorship of the Office of Secretary of Defense (Force Management and Personnel). The technical sporsor was Mr. Gary Boycan. Funding was provided under Program Element 0604722J, Work Unit 99-PJ1-90-006.

This document was developed under the technical supervision of Dr. Raye Newmen, Dr. Wallace H. Wulfeck, and Mr. Walter F. Thode of the Navy Personnel Research and Development Center (NPRDC). This report itself was written by Systems Engineering Associates under Contract N66001-88-D-0054, Delivery Order 7J30.

This architecture was developed as part of an effort to complete a reference implementation of a courseware portability specification. The implementation is scheduled for later this year. Comments on the architecture described here are solicited from all interested parties. The specification will then be submitted for consideration for official adoption by the Department of Defense and the National Institute of Standards and Technology.

Point of contact at NPRDC is Walter F. Thode (619) 553-7703 or AUTOVON 553-7703.

WALLACE H. WULFECK II
Director, Training Technology Department

Summary

Background

Over the next several years the Federal Government will invest millions of dollars to develop training materials for delivery on computer-based interactive training systems. To support this investment, Federal agencies will acquire a variety of computers and peripheral devices. This hardware will host several operating systems and authoring system software to speed courseware development.

Many vendors produce high-performance, low-cost training hardware, but bundle their products with proprietary software interfaces. Because these interfaces are proprietary, courseware and authoring systems written to operate on one set of hardware will not run on a competitor's hardware. Expensive reprogramming is needed to adapt to new hardware. These reprogramming costs can be eliminated by adopting standard software interfaces.

Objectives

The objectives of this effort were to describe and develop a standard software interface that will allow training systems to be assembled from separate "plug-and-play" components in the same way that stereo systems can be assembled from separate speakers, amplifiers, and other components.

Approach

The Portable Courseware (PORTCO) architecture consists of two interfaces, the Device Services Interface and the Device Handler Interface. It also contains three layers: application, routing and configuration, and the device handler. This architecture should allow applications software to run on any compliant set of hardware components.

Results and Conclusions

This report is the first in a series of five reports; it provides an overview of the PORTCO architecture and should be of interest to all who are concerned with computer-based training. The second report describes the MS-DOS Device Services Interface and is intended primarily for programmers who want to develop portable application software. The third report describes the Device Handler Interface and should be of primary interest to device manufacturers and system vendors who must develop device handler software. The fourth report is intended for system vendors and describes the design

of the first PORTCO routing and configuration program. The fifth report is intended as additonal support for those who must develop complaint MS-DOS device handlers.

Recommendations

1. The reports describing the PORTCO architecture should direct development of portable MS-DOS applications and standard peripheral device handlers.

2. The architecture should serve as a foundation for compliance with the Interactive Video Industry Association's "Recommended Practices for Interactive Video Portability." The architecture should motivate development of specific applications and device handlers that adhere to this specification.

3. Feedback about problems and suggested improvements should be forwarded to the Navy Personnel Research and Development Center, Code 152.

Table of Contents

List of Figures

1. Introduction

Over the next several years the Federal Government will invest millions of dollars to develop training materials for delivery on computer-based interactive training systems. To support this investment, Federal Agencies will acquire a variety of computers and *peripheral devices.*[1] This hardware will host MS-DOS[2] or UNIX[3] operating systems and *authoring system* software to speed *courseware* development.

Many vendors produce high-performance, low-cost training hardware, but bundle their products with proprietary *software interfaces*. Vendors of *integrated training systems* bundle their products with interfaces to all system resources, while vendors of individual peripherals (such as mice) bundle their products with simpler, single-device interfaces. Because these interfaces are proprietary, courseware and authoring software written to operate on one product will not run on a competitor's product. Expensive reprogramming is needed to adapt to new peripheral hardware. These reprogramming costs can be eliminated by adopting standard software interfaces.

This paper introduces two standard interfaces that allow training systems to be assembled from separate "plug-and-play" components in the same way that stereo systems can be assembled from separate speakers, amplifiers, etc. Named the *Device Services Interface (DSI)* and the *Device Handler Interface (DHI)*, they are part of the *Portable Courseware (PORTCO) architecture* illustrated in figure 1 (and discussed in section 4).

The DSI allows *application* authors and *system integrators* to write software that controls generic *logical devices*, instead of particular devices from particular manufacturers. The DHI allows vendors to write standard *device handlers*. Together, the DSI and DHI provide a foundation to build sophisticated applications that can easily adapt to changes in individual system components.

1. Special terms appear in *italics* when first used, and are defined in section 6.
2. MS-DOS is a registered trademark of Microsoft Corporation.
3. UNIX is a trademark of AT&T Bell Laboratories.

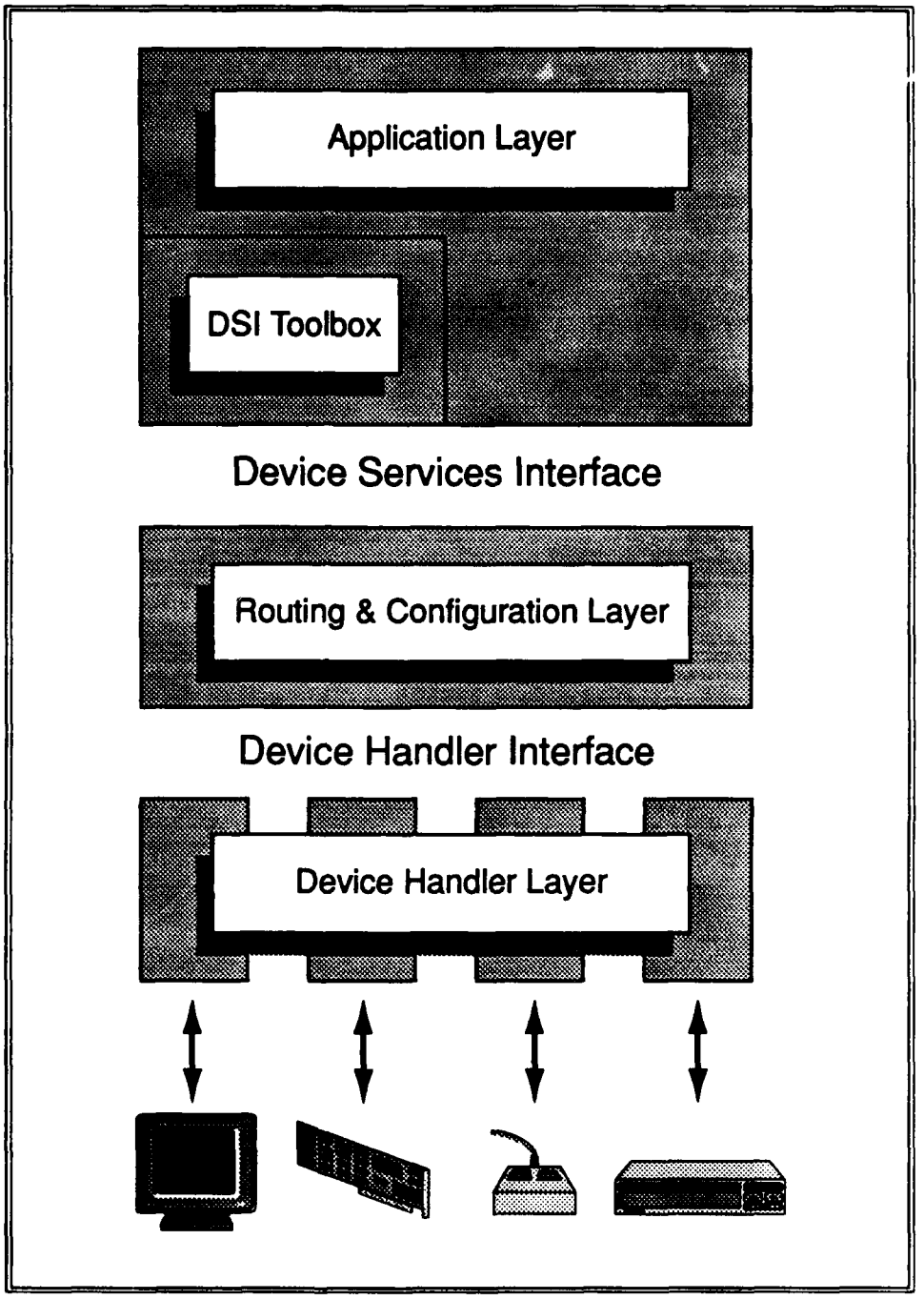


Figure 1: A three-layer PORTCO architecture will enhance portability.

2. Purpose and Scope

This paper describes the PORTCO architecture and its two standard software interfaces, the DSI and the DHI. The PORTCO architecture will be implemented in two stages, first on MS-DOS-based platforms and then on UNIX-based platforms. This is one of several publications guiding the architecture's MS-DOS implementation. Section 7 lists other MS-DOS-based PORTCO publications. A similar set of documents will be published in FY 90 depicting the architecture's UNIX implementation.

This paper provides an overview of the PORTCO architecture; details are provided in other PORTCO publications. This paper should be reviewed by all users, administrators, developers, and maintainers before reading the other publications listed in section 7.

The following section introduces logical devices and *device classes*, and shows how these concepts can be applied to insulate *application software* from changes in peripheral equipment. Section 4 presents the three-layer PORTCO architecture. It describes the purpose of each *layer*, provides an overview of the layers' packet-based communication *protocols*, and describes the architecture's two standard interfaces. For readers who desire more information, section 5 elaborates upon the other PORTCO publications and helps direct further reading. Section 6 presents a glossary of special words used in this paper.

3. Preliminary Concepts

Peripheral devices from different vendors frequently share common features. But to distinguish themselves in the marketplace, vendors add product features and protocols that are slightly different from their competitors.. Application authors must create expensive new software to accommodate these similar, yet different, peripheral protocols and services.

For example, different vendors might design their videodisc players to display a still frame in response to different command strings. Or one vendor might provide separate commands to present a video motion sequence ("Set End Marker," "Search To Start Frame," "Play To End Marker"), while another might provide a single high-level command to play from one video frame to another. Accommodating these differences requires custom programming for each player. Such programming can be eliminated by writing software to control logical devices instead of particular peripherals from particular vendors.

3.1 Logical Devices and Device Classes

A logical device is a concept (not a piece of hardware) that is synthesized from characteristics of several similar peripherals. For example, a logical videodisc player might be synthesized with attributes common to many actual players: It may have a door to insert or remove videodiscs; it may have a remote control unit to usurp computer control; and it may respond to a fixed collection of display and action command strings.

Because a logical device is a concept, not a piece of hardware, it is characterized only by the services it provides, not by its physical attributes. A collection of logical device services is called a device class.

A device class is like a mold from which identical logical devices are fashioned. Just as a mold used to cast an automobile engine block precisely defines the size and shape of the block, services in a device class precisely define the behavior of a logical device. For example, services in the videodisc player device class precisely define the behavior of a logical videodisc player. (All PORTCO device classes are specified in references 1 and 2).

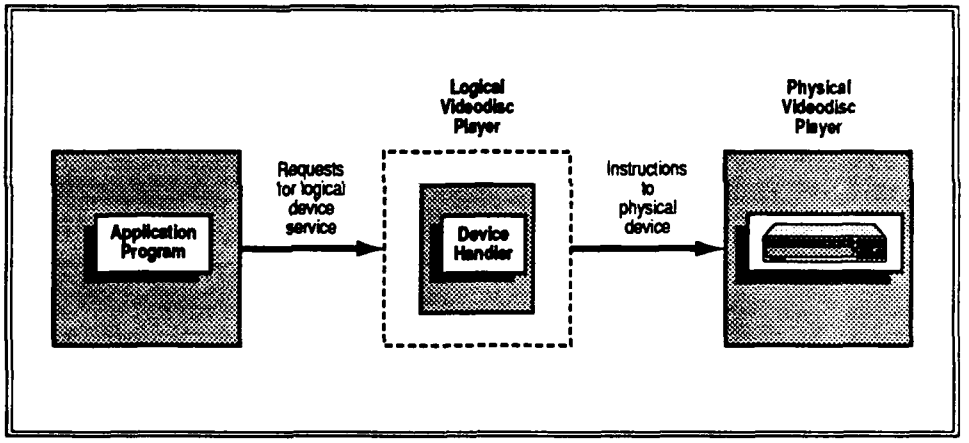
When an application author designs software to control a logical device, he does not need to be concerned with the operation of any particular vendor's peripheral. The author must only be concerned with the

services provided by the logical device (i.e., the services in its device class). So using logical devices insulates an author from the details of any particular peripheral's operation.

3.2 Device Handlers

To run a program designed for logical devices on real hardware, part of the system must translate requests for logical device service into instructions that real hardware understands. *Software modules* that perform this translation are called device handlers. Figure 2 illustrates the relationship between a logical device, a *physical device*, and a device handler. In this figure, an application issues service requests to a logical videodisc player. A device handler intercepts these requests and translates them into instructions that a physical videodisc player can understand.

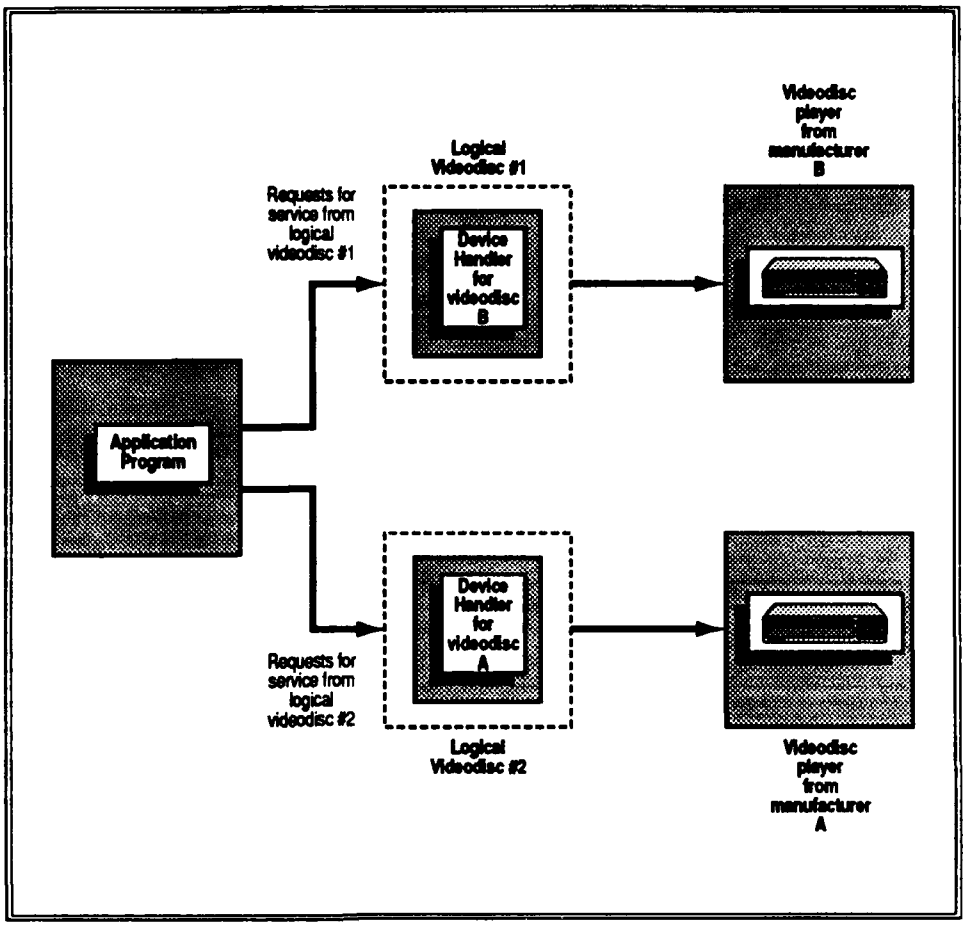
Figure 2: A device handler translates requests for logical device service into instructions that a physical device can understand.



Identical peripherals from the same manufacturer can be interchanged without affecting an application's performance because they have identical hardware and software interfaces. Such peripherals are like identical twins, they "look" the same to applications. To plug-and-play peripherals from different manufacturers, these different peripherals must also "look" identical to applications. Standard device handlers can make dissimilar peripherals "look" the same.

For example, because the application in figure 2 requests service from a logical videodisc player, not from the physical peripheral, it "sees" only the player's standard device handler, not the player itself. A player from any other manufacturer can therefore be substituted, without changing the application, by at the same time substituting a different device handler.

Figure 3: Applications request service from multiple logical devices by specifying a device's class and number.



Logical devices and standard device handlers can also insulate from peripheral hardware those applications that control two or more devices. For example, figure 3 shows a program that controls two videodisc players, one from manufacturer A and one from manufacturer B. The program requests service by specifying the applicable device class (videodisc player) and the logical device from that class (player #1 or player #2). As in figure 2, either of the players in figure 3 might be replaced with players of different makes and models by changing the device handler, not the application.

3.3 Core and Extended Services

Use of logical devices and standard device handlers lets applications request peripheral services in a standard way, but some physical peripherals may not support all services in their device class. For example, the PORTCO overlay device class contains a service to set a video dissolve level, but not all overlay cards can effect a video dissolve.

To solve this problem, each PORTCO device class identifies a subset of services that are common to all physical devices in the class. These are called *core services*. The rest are called *extended services*, and may or may not be available on a specific physical peripheral in the class. For example, the service described above to set a video dissolve level would be an extended service (because it is not available on all overlay cards), while a service to make a color transparent and opaque would be a core service (because all overlay cards offer it).

Application authors can ensure portability by using core services. Using extended services may enhance performance, but will reduce portability to only those peripherals that support the required extensions. So extended services allow application authors to balance their products' portability and performance. They also allow vendors to differentiate themselves in the marketplace by letting them choose which extensions (if any) their products will offer.

The PORTCO architecture must evolve gracefully with advancing technology. Core services, extended services, and device classes provide a framework for this evolution. Extended services will be added frequently as manufacturers create new hardware features. These will become core services as they are adopted by a larger segment of the peripheral marketplace. New device classes will be added with significant breakthroughs in peripheral technology (e.g., digital video interactive).

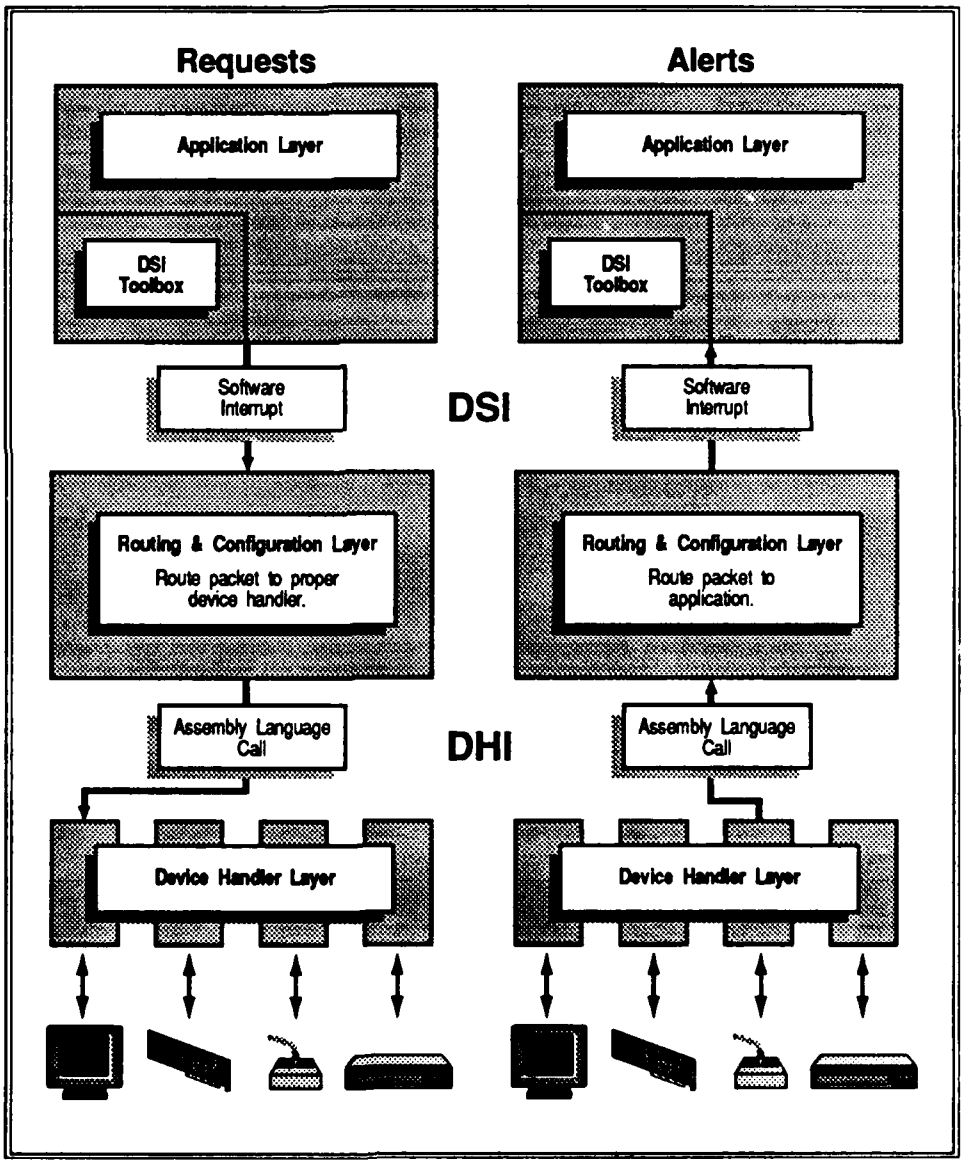
3.4 Requests and Alerts

Applications must be able to request services from logical devices, and they must be alerted asynchronously when important device activity occurs. For example, an application should be able to ask a videodisc player to show a still frame, and it should be alerted when a mouse movement or button-press occurs. Two types of services are included in each PORTCO device class to satisfy this requirement. *Requests* are used to solicit service, and *alerts* are used to inform applications of asynchronous device activity. All requests and alerts are specified in references 1 and 2.

4. PORTCO Architecture

The PORTCO architecture contains three layers and two standard interfaces, illustrated in figure 1 (page 2). Each layer communicates with its neighbor(s) using contiguous blocks of data called *packets*. For example, applications request peripheral service by passing packets through the *routing and configuration layer* (*R&C layer*) to device handlers. Figure 4 illustrates packet routing for PORTCO requests and alerts.

Figure 4: Routing of request and alert packets is almost identical.



Standard device handlers constitute the lowest layer of the PORTCO architecture. They translate requests for logical device service into instructions that real peripherals can understand. They also alert the architecture's upper layers when device activity (e.g., a mouse movement) occurs. As discussed in section 3.2, device handlers are the only PORTCO components that depend upon physical devices; they are the only components that must be replaced when peripheral devices are changed. Device handlers are implemented as modified MS-DOS executable files.

The middle layer of the PORTCO architecture is called the R&C layer. It loads and configures device handlers and routes packets between applications and device handlers. For example, in the system illustrated by figure 3 (page 6), the R&C layer would direct all packets requesting service from logical videodisc player #2 to videodisc player A's device handler (instead of videodisc player B's). The R&C layer is implemented as a *TSR program*.

Application software constitutes the PORTCO architecture's highest partition. This layer also contains the *DSI toolbox*. This is a collection of software modules (developed by the PORTCO user community) that helps applications use the DSI. In the simplest case, the toolbox consists of functions in various programming languages that format packets and pass them to the R&C layer (i.e., language bindings). More sophisticated toolbox components might coordinate the activities of several logical devices (e.g., to track a logical mouse's cursor on a logical graphics device) or implement more abstract virtual devices (e.g., an integrated videodisc system).

4.1 Standard Interfaces

Boundaries between adjacent layers of the PORTCO architecture are called interfaces. The boundary between the application layer and the R&C layer is called the DSI, and the boundary between the R&C layer and device handlers is called the DHI. Each of these interfaces consists of a collection of services and a set of protocols (or rules) for invoking them. For example, one of the services offered by the DSI asks a logical videodisc to display a still frame. Applications employ a packet-passing protocol (using a software interrupt) to invoke this service.

DSI services are used by applications and the DSI toolbox to control logical devices. DHI services are used by the R&C layer to load, configure, and control device handlers. The DSI guides construction of applications and application interfaces, while the DHI guides construction of device handlers.

Services offered by the DSI and the DHI are substantially the same, and are defined by the PORTCO architecture's device classes. When an application uses the DSI to manipulate a logical device, the R&C layer routes the request to the proper device handler through the DHI. Likewise, when a device handler alerts the R&C layer (via the DHI) of device activity, the R&C layer routes this alert to the application (via the DSI).

The DSI and DHI are not identical. Protocols used by each interface to invoke services differ, and each interface contains a unique set of initialization and configuration services. References 1 and 2 detail the protocols and services used by each interface.

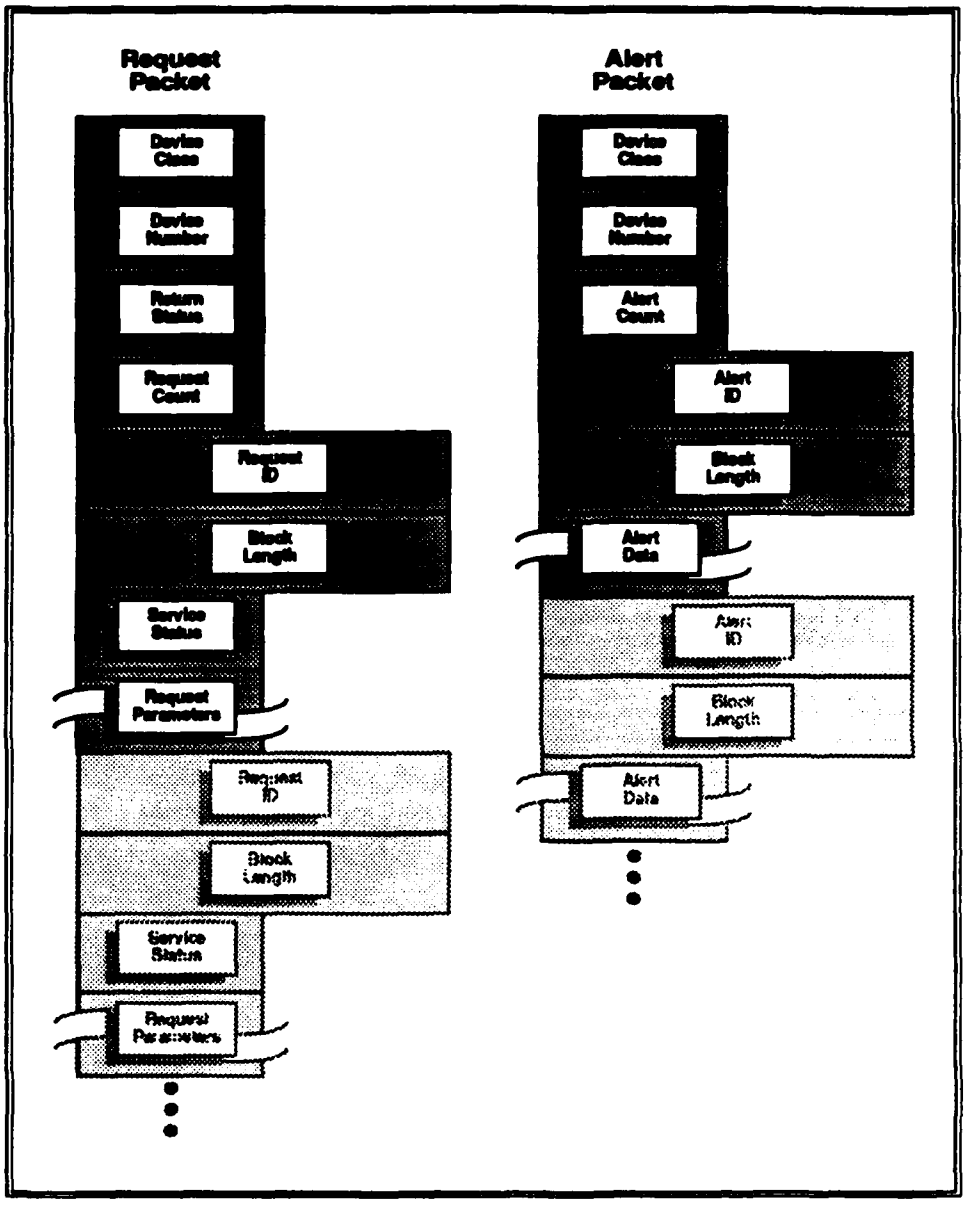
The DSI toolbox and application software also share a boundary, as shown in figure 1. This means that the toolbox can offer its own interface to applications. This interface may range from an informal set of function calls produced by a particular vendor for its own use, to a standard like the Interactive Video Industry Association's (IVIA's) Application Interface. In either case, all software in the toolbox employs the DSI to communicate with peripheral devices, and is thus portable to any other PORTCO-compliant system. For example, implementing the IVIA's Application Interface as a DSI toolbox component makes this standard immediately available on all PORTCO-compliant systems.

4.2 Packets

Adjacent layers in the PORTCO architecture communicate using contiguous blocks of data called packets. Two types of packets are used: *request packets*, to solicit services, and *alert packets*, to announce device activity.

Figure 5 illustrates each type of packet. As this figure suggests, each request packet identifies one or more services and a logical device that should perform them. Each alert packet identifies one or more activities completed asynchronously by a logical device. Logical devices are described by their device class and device number, and services are specified using a service number and various parametric data.

Figure 5: Request packets communicate one or more service requests; alert packets report asynchronous device activity.



As illustrated in figure 4 (page 8), packets are passed between the application layer and the R&C layer using a software interrupt, and between the R&C layer and device handlers using an assembly language call. Details concerning inter-layer communication and packet formats appear in references 1 and 2.

4.3 Conformance

Conformance to the PORTCO architecture implies use of the DSI and the DHI. Applications and toolbox components conform by using the DSI. R&C-layer software conforms by implementing the DSI using the DHI. Device handlers conform by implementing a single device class in the DHI.

Applications and toolbox components that use the DHI directly do not conform.

5. Further Reading

Application authors, system administrators, and readers intending to build R&C-layer software should study reference 1, "The MS-DOS Device Services Interface." This document describes each DSI service in detail, and specifies each service's packet format and invocation protocol. It also designates core and extended services, and describes the DSI's initialization and configuration services.

Device handler and R&C-layer developers should review reference 2, "The MS-DOS Device Handler Interface." This document describes all DHI services and their invocation protocols. It also discusses the R&C layer's role in loading and initializing device handlers, and it provides detailed constraints for developing compliant handlers.

R&C-layer and device-handler developers should also review references 3 and 4, "MS-DOS Routing and Configuration Layer Program Design Document" and "Guidelines for Implementing MS-DOS PORTCO Device Handlers." These documents present detailed examples of software that implements the PORTCO architecture.

6. Glossary

The following list defines terms used in this paper. These terms are defined in the context of their use in this paper and the other documents listed in section 7. The definitions may therefore be more restrictive than, or otherwise differ from, the commonly accepted definitions. Words that are italicized in definitions are themselves defined elsewhere in this list.

Alert — A service in the *DSI* that interrupts an *application* when a specific *peripheral device* activity occurs. A service in the *DHI* that interrupts the *R&C Layer* when a specific *peripheral device* activity occurs.

Alert packet — A *packet* used to communicate an *alert* between *layers* of the *PORTCO architecture*.

Application — *Application software*.

Application layer — The highest partition in the *PORTCO architecture*. Contains the *DSI toolbox*.

Application software — Any software that is part of the *application layer*.

Architecture — The organization or structure of a system or of a system component.

Authoring system — Software that aids the development of computer-based instruction.

Core service — A service within a *device class* that must be provided by every compliant *device handler* in the class.

Courseware — Software and/or data used to present computer-based instruction.

Device class — A collection of services provided by a single *logical device*.

Device handler — A *software module* that implements the *DHI* for a single *device class*. Translates requests for service from a *logical device* into instructions that a physical peripheral can understand.

Device handler layer	The lowest *layer* of the *PORTCO architecture*. Contains *device handlers*.
Device Handler Interface (DHI)	A standard collection of services and *protocols* that defines the boundary between the *R&C layer* and the *device handler layer*.
Device Services Interface (DSI)	A standard collection of services and *protocols* that defines the boundary between the *application layer* and the *R&C layer*.
DSI toolbox	A *software module* (or modules) providing an *interface* between the *application software* and the *R&C layer*.
Extended service	A service within a *device class* that may be provided by compliant *device handlers* in the class.
Integrated system	A collection of computer hardware and software sold as a single unit by a *system integrator*.
Interface	A *software interface*.
Layer	A group of related functions that make up one level of a *layered architecture*.
Layered architecture	A *software architecture* in which components are grouped in a hierarchical arrangement in such a way that each *layer* provides functions and services to adjacent *layers*.
Logical device	A conceptual device synthesized by using characteristics of several similar *peripherals*. Specified by a *device class*.
Packet	A contiguous block of data used to communicate information between *layers* of the *PORTCO architecture*.
Peripheral device	Any *physical device* that is distinct from a computer's main processor.
Physical device	A computer system hardware component.

Portability	The ability of *applications* to run correctly on various *system configurations*.
PORTCO architecture	A *layered architecture* presenting a standard *interface* between *peripheral devices* and *applications*.
Protocol	A set of rules governing the communication between two *software modules*.
R&C layer	*Routing and configuration layer*.
Request	A service in the *DSI* or the *DHI* that demands action from a *peripheral*.
Request packet	A *packet* used to communicate a *request* between *layers* of the *PORTCO architecture*.
Routing and configuration layer	The middle partition of the *PORTCO architecture*.
Software interface	The boundary between two or more *software modules*, or a *protocol* that defines how two *software modules* communicate.
Software module	A named collection of software instructions and data.
System configuration	The number and types of *peripheral devices* connected to a computer.
System integrator	A vendor that assembles and sells an *integrated system*.
TSR program	An MS-DOS terminate-and-stay-resident program.

7. References

1. Thomason, B. L., Van de Wetering, B. L., *The MS-DOS Device Services Interface*. Systems Engineering Associates, San Diego, CA, Feb. 26, 1990.

2. Van de Wetering, B. L., Thomason, B. L., *The MS-DOS Device Handler Interface*. Systems Engineering Associates, San Diego, CA, Feb. 26, 1990.

3. Van de Wetering, B. L., Thomason, B. L., *The MS-DOS Routing and Configuration Program Design*. Systems Engineering Associates, San Diego, CA, Feb. 26, 1990.

4. Van de Wetering, B. L., Thomason, B. L., *Guidelines for Implementing MS-DOS PORTCO Device Handlers*. Systems Engineering Associates, San Diego, CA, to be issued August 1990.

Distribution List

Distribution:
Assistant Secretary of Defense (Force Management and Personnel)
Deputy Under Secretary of Defense for Research and Engineering (Research and Advanced Technology)
Director, Total Force Training and Education (OP-11)
Director, Aviation Training Systems Program Coordinator (PMA-205)
Commanding Officer, Naval Training Systems Center
Defense Technical Information Center (DTIC) (2)

Copy to:
Deputy Chief of Naval Operations (MP&T) (OP-01)
Assistant for Planning and Technical Development (OP-01B2)
Head, Training and Education Assessment (OP-11B1)
Director, Total Force Information System Management (OP-16)
Director, Submarine Manpower and Training Requirements (OP-29)
Director, Command Surface Warfare Manpower and Training Requirements (OP-39)
Director, Air ASW Training (OP594)
Assistant for Manpower, Personnel, and Training (OP-983D)
Commander, Naval Sea Systems Command (PMS 350)
Commander, Naval Sea Systems Command (PMS 396)
Commander, Naval Sea Systems Command (CEL-MP)
Director, Strategic Systems Project (SP-15)
Commanding Officer, New London Laboratory, Naval Underwater Systems Center (Code 33A)
Commanding Officer, New London Laboratory, Naval Underwater Systems Center (Code 3333)
Naval Training Systems Center, Technical Library (5)
Naval Training Systems Center (Code 10), (Code N-1), (Code 7)
Director, Office of Naval Research (OCNR-10)
Chief Scientist, Office of Naval Technology (OCNR-20T)
Chief of Naval Education and Training (Code 00)
Director, Training Technology (Code N-54)
Commanding Officer, Naval Education and Training Program Management Support Activity (Code 03) (2)
Commanding Officer, Naval Education and Training Program Management Support Activity (Code 04) (2)
Chief of Naval Technical Training (Code 00) (2)
Commander, Naval Military Personnel Command (NMPC-00/PERS-1)
Naval Military Personnel Command, Library (Code NMPC-013D)
Commanding Officer, Naval Health Sciences Education and Training Command, Bethesda, MD
Commander, Naval Reserve Force, New Orleans, LA
Commandant of the Marine Corps, Commanding General, Marine Corps Research and Development and Acquisition Command
Commander, U.S. ARI, Behavioral and Social Sciences, Alexandria, VA (PERTI-POT-I)
Technical Director, U.S. ARI, Behavioral and Social Sciences, Alexandria, VA (PERI-ZT)
Commander, Air Force Human Resources Laboratory, Brooks Air Force Base, TX

Scientific and Technical Information (STINFO) Office
TSRL/Technical Library (FL 2870)
Program Manager, Life Sciences Directorate, Bolling Air Force Base, DC (AFOSR/NL)
Commander, OPSTNGDIV Air Force Human Resources Laboratory, Williams Air Force Base, AZ (AFHRL/OT)
Commander, Air Force Human Resources Laboratory, Wright-Patterson Air Force Base, OH, Logistics and Human Factors Division (AFHRL/LRS-TDC)
Director of Training, Office of Civilian Personnel Management
Superintendent, Naval Postgraduate School
Director of Research, U.S. Naval Academy
Center for Naval Analyses