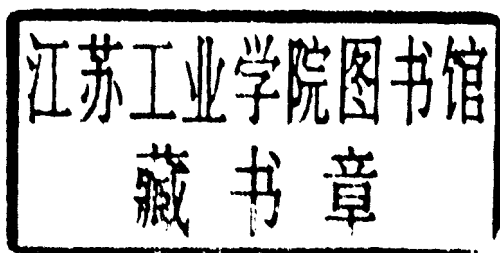




**Programming Practices
and Standards**
Application Design

CTOS/Open
Programming Practices and Standards
Application Design



PRENTICE HALL
Englewood Cliffs, New Jersey 07632

Authors: *ALAN COLEMAN* and *MARGARET MORRIS*
Cover design: *APRIL BISHOP*
Page design: *MILENA MARTIN-ARANA*
Editorial/production supervision: *MARY ROTTINO*
Manufacturing buyers: *KELLY BEHR* and *SUSAN BRUNKE*
Acquisitions editor: *KAREN GETTMAN*

© 1991, 1990 by *Convergent, Inc.*



Published by Prentice-Hall, Inc.
A Division of Simon & Schuster
Englewood Cliffs, New Jersey 07632

Convergent, Convergent Technologies, NGEN, and CTOS are registered trademarks of Convergent, Inc.

Context Manager, CTOS, CTOS/VM, Generic Print System, Shared Resource Processor, SRP, The Cluster, and X-Bus are trademarks of Convergent, Inc.

The publisher offers discounts on this book when ordered in bulk quantities. For more information, write:

Special Sales/College Marketing
Prentice-Hall, Inc.
College Technical and Reference Division
Prentice Hall
Englewood Cliffs, New Jersey 07632

All rights reserved. No part of this book may be reproduced, transmitted, stored in a retrieval system, or translated into any language without permission in writing from Convergent, Inc. or the publisher.

Printed in the United States of America
10 9 8 7 6 5 4 3 2 1

ISBN 0-13-194382-0

Prentice-Hall International (UK) Limited, *London*
Prentice-Hall of Australia Pty. Limited, *Sydney*
Prentice-Hall Canada Inc., *Toronto*
Prentice-Hall Hispanoamericana, S.A., *Mexico*
Prentice-Hall of India Private Limited, *New Delhi*
Prentice-Hall of Japan, Inc., *Tokyo*
Simon & Schuster Asia Pte. Ltd., *Singapore*
Editora Prentice-Hall do Brasil, Ltda., *Rio de Janeiro*

About This Manual

This manual describes recommended techniques and practices for writing portable applications that follow the CTOS/Open standard for the CTOS® operating system. This manual is a companion volume to the *CTOS/Open Application Programming Interface Specification*.

The CTOS/Open standard is a specification which is implemented by several CTOS-based operating systems. The standard is not itself an operating system. When this manual talks about CTOS/Open, it means the specification. When it talks about CTOS, or about CTOS-based operating systems, it means those operating systems that conform to the CTOS/Open standard.

The CTOS/Open standard is a common subset of features. The operating systems that comply with it generally have additional features beyond the ones included in the CTOS/Open standard.

Purpose of This Manual

This manual describes practices and standards programmers should follow to ensure that their applications are portable across the various implementations of the CTOS/Open standard. Following the recommendations in the standard can greatly simplify porting an application from one CTOS-based operating system to another.

This manual can also serve as a "programmer's primer" for those new to CTOS/Open. It provides many examples, showing how to write commonly-used operations. This manual does not provide an in-depth conceptual explanation of CTOS implementation internals. For more comprehensive conceptual information, see the manuals for the CTOS-based operating system you use.

Audience

This manual is intended for programmers who want to write portable applications for CTOS-based operating systems. Programmers who are experienced with these operating systems, and programmers new to them, should both find the recommendations and examples in this book useful.

Related Documentation

The following documents may be useful to programmers who read this book.

CTOS/Open Books

- *CTOS/Open Application Programming Interface Specification*

This document describes the procedural interface definitions for each of the procedures supported by the CTOS/Open Standard. It defines a common set of system interfaces that is consistent across all CTOS/Open-compliant versions of CTOS. The specification is intended for use by software developers and Independent Software Vendors (ISVs) who want to write applications that conform to the CTOS/Open Standard and are therefore portable across vendor platforms. It assumes that the reader has experience writing applications under CTOS or under another operating system.

- *CTOS/Open Application Programming Interface Specification: Computer Graphics Interface (CGI)*

This document describes each CGI operation and explains how to write your own CGI programs.

- *CTOS/Open Application Programming Interface Specification: Graphical User Interface (GUI)*

This document introduces an eXtensible Virtual Toolkit (XVT) and describes its relationship to other windowing interfaces. It also reviews the current XVT operation set, which includes functions,

macros, constants, and types. This specification is a virtual API and is intended for programmers who want to write applications that run in several different window environments and work across all CTOS-based platforms.

- *CTOS/Open Application Programming Interface Specification: Networking Services*

This document describes the standard Link Layer interface used under the CTOS operating system. It defines the common requests used by all Link Layers and the requests used by the Data Link Manager. This specification also provides the standard format for reporting Link Layer status, and documents Link Layer error messages/codes, event codes, and commands. The final draft of this manual will include information on the Transport Layer.

- *CTOS/Open Application Programming Interface Specification: Printing Services (GPS)*

This document describes how to write applications that use the Generic Print System (GPS) or the Generic Print Access Method (GPAM).

- *CTOS/Open Programming Practices and Standards: User Interface Design*

This document provides guidelines for designing graphical user interfaces for CTOS-based applications. It contains information about the user interface standard call Common User Access (CUA), which has become part of the public domain. This manual is a working draft.

- *Exploring CTOS*

This book gives a conceptual overview of the CTOS architecture. Additionally, it explains the advantages of developing applications on a distributed, message-based operating system.

Intel Manuals

- *iAPX 286 Programmer's Reference Manual*

This book describes the features and instruction set of the Intel 80286 microprocessor.

- *80386 Programmer's Reference Manual*

This book describes the features and instruction set of the Intel 80386 microprocessor.

How to Use This Manual

This manual is organized by function. Individual chapters are devoted to each of the most commonly-used functional areas of CTOS. To use this manual, examine the chapter or chapters that apply to your project, and make use of the examples and recommendations you find there. To look up specific topics, use the index at the back of the manual.

The CTOS Naming Convention

The examples in this book follow a specific naming convention, which is designed to promote the readability of source code. The underlying principal of the naming convention is this: use explanatory prefixes and suffixes on all variables and procedure names. Elementary programming practice dictates that the names themselves should also be explanatory. These conventions are particularly important when programming in a language like C, which tends toward cryptic syntax.

Each variable takes the form <Prefix><Root>Name<Suffix>. Prefix, root, and suffix do not all have to be present. In fact, most variables use only a prefix and a root in combination with the variable name itself. The following table describes the CTOS naming convention.

Table ATM-1. CTOS Variable-Naming Convention

(Page 1 of 2)

Token	Meaning
PREFIXES:	
b	Byte. A character or unsigned 8-bit integer.
c	Count. A two-byte unsigned integer.
f	Flag. A one-byte flag. True = 0FFh, False = 0.
i	Index. A two-byte unsigned integer.
l	Literal. A constant.
n	Number. A two-byte unsigned integer. Same as Count.
o	Offset. A two-byte offset from a segment base address.
p	Pointer. A logical memory address. Consists of a two byte segment identifier and a two-byte offset.
q	Quad. A four-byte unsigned integer.
rg	Array. Usually used with another prefix. For example, the prefix "rgb" identifies an array of bytes.
s	Size. A two-byte unsigned integer.
sb	String. An array of bytes where the first byte is the size of the string.
w	Word. A two-byte integer.
cb	Count of bytes.
pb	Pointer to a string of bytes.

Table ATM-1. CTOS Variable-Naming Convention

(Page 2 of 2)

Token	Meaning
ROOTS:	
erc	Two-byte status code.
exch	Two-byte exchange number.
fh	Two-byte file handle.
lfa	Four-byte logical file address.
ra	Two-byte relative address. Synonymous with offset.
rq	Request block. Size varies.
sa	Two-byte segment identifier.
sn	Selector. Segment identifier for a protected-mode memory address.
sr	Paragraph number. Segment identifier for a real-mode memory address.
userNum	Two-byte user number.
SUFFIXES:	
Last	Largest allowable index in an array.
Max	Maximum size of an array or buffer (Max = Last + 1).
Ret	Indicates a variable whose value is set by a called procedure and returned to the current one.

For example, to define a data buffer using this naming convention, we can assign four variable names:

`pBuffer`. A pointer to the start of the buffer.

`sBufferMax`. The maximum size of the buffer.

`sBufferDataRet`. The size of the data actually written to the buffer, returned by the procedure that writes to the buffer.

`psBufferDataRet`. Address of `sBufferDataRet`. Passed to the procedure that writes to the buffer, telling that procedure where to return the value of `sBufferDataRet`.

Part I – The Essentials

About This Manual

Purpose of This Manual	xxiii
Audience	xxiv
Related Documentation	xxiv
How to Use This Manual	xxvi
The CTOS Naming Convention	xxvi

Part I - The Essentials

1 CTOS Overview

What Is CTOS/Open?	1-1
The Major Features of CTOS	1-1
Multiprogramming	1-2
Multitasking	1-2
Process Scheduling	1-2
Message-Based Operation	1-3
Extensible Via System Services	1-4
Nationalization	1-4
Important Concepts	1-4
Processes, Messages and Exchanges	1-4
The Request/Response Model	1-6
Programs and Partitions	1-7
The Distributed Environment	1-8
System Memory Organization	1-8
Application Memory Organization	1-9
The Anatomy of a Program	1-11
Segmentation	1-11

	Types of External Procedures, and How to Call Them	1-11
	A Note About pbcbs	1-13
	What Happens When a Program Starts	1-14
	What Happens When a Program Exits	1-14
2	Basic Input and Output	
	Device-Independent I/O	2-1
	The Sequential Access Method (SAM)	2-1
	What Is a Byte Stream?	2-2
	Supported Byte Streams	2-2
	Device and File Specification Parsing	2-4
	Using a Generic Byte Stream	2-5
	Using Byte Streams for File Access	2-8
	Using Byte Streams for Video Access	2-9
	Using Byte Streams for Keyboard Access	2-12
	Other Uses for Byte Streams	2-14
	Device-Specific I/O	2-14
	File Management	2-14
	Wild Cards	2-16
	Temporary Files	2-17
	Keyboard Management	2-18
	Video Management - VAM and VDM	2-20
3	Error Handling Conventions	
	Error Checking: General Practice	3-1
	The FatalError Procedure and the fDevelopment Flag	3-2
	Program Exit Modes: Exit, ErrorExit, ErrorExitString, and Crash	3-2
	Trapping Protection Faults	3-3
	Error Logging: WriteLog	3-6
4	Parameters and Command Form Processing	
	The Executive	4-1
	Command Forms and Parameters	4-1
	The Variable-Length Parameter Block (VLPB)	4-2
	Reading Input Parameters from the VLPB	4-4
	Creating a New VLPB for the Exit Run File	4-6

5	Protected Mode Programming Guidelines	
	Real Mode and Protected Mode Compatibility	5-1
	Review of Segmented Addressing	5-2
	Real Mode versus Protected Mode Pointers.....	5-2
	General Programming Guidelines	5-3
	Language-Specific Guidelines	5-11
	Real Mode Guidelines	5-12
6	Writing Your Application for International Use	
	Introduction	6-1
	Using the NLS Tables and System Calls	6-2
	Using the NLS Procedures.....	6-4
	Using the NLS Tables.....	6-5
	Using Alternative NLS Tables	6-7
	Linking Alternative Tables with Your Program	6-7
	Using Additional NLS Tables.....	6-8
	Using Message Files.....	6-8
	About Message Files.....	6-8
	Strategies for Using Messages	6-9
	Using Messages as Needed.....	6-10
	Standard Message Routines	6-11
	Message File Macros	6-13
	Pre-Loading Messages.....	6-15
	Using a Very Small Number of Messages	6-16
7	Tips for the Application Writer	
	Introduction	7-1
	Program Structure and Design	7-1
	Keyboard-Handling Conventions	7-3
	Application-Independent Key Meanings and Their Use	7-3
	Application-Dependent Key Use Conventions	7-4
	Keyboard Events an Application Must Handle.....	7-6
	Screen Layout Conventions	7-6
	Function Key Menus.....	7-6
	Help	7-7
	Status Information	7-8
	Creating An Executive Screen	7-8

Cleanliness	7-9
Guidelines for Screen Handling	7-9
File Suffix Conventions	7-10
The Scratch Volume	7-11
 8 Writing Request-Based System Services	
Introduction	8-1
Requests and Request Levels	8-1
Registering Request Codes	8-4
By Telephone	8-4
By FAX	8-5
Conserving Request Codes and Memory	8-5
Conserving Memory with Efficient Request Code Use	8-6
Conserving Request Code Numbers	8-6
Operations Performed by a System Service for Applications ...	8-7
Server/Client Communication	8-7
The Server/Client Relationship	8-8
Connection Establishment	8-8
A Note About Handles	8-10
The Real Work	8-11
Connection Termination	8-12
Connectionless Requests	8-13
Operations Performed by a System Service for the	
Operating System	8-13
Termination and Abort Requests	8-14
Swapping Requests	8-16
Installation and Deinstallation	8-17
Installation	8-17
Deinstallation	8-20
Deinstallation on Error	8-22
System Services that Act as Filters	8-23
Types of Filters	8-23
Requirements for Filters	8-24
Note on Keyboard Filters	8-25
System Services that Act as Agents	8-25
Role of the Client Agent	8-26
Role of the Server Agent	8-27
Request-Passing Guidelines for Agents	8-28

Piecemealing of Very Large Request Blocks	8-28
Defining Request Codes for a System Service	8-30
What You Need to Define	8-30
Defining a Request	8-31
The Structure of a Request	8-32
Defining the Procedural Interface	8-34
Defining Request Routing	8-35
Making a Loadable Request Set	8-39
Making a Request Label Object File	8-40
 9 Writing System-Common Services	
Introduction	9-1
Request-Based vs. System-Common Services	9-1
The System-Common Model	9-1
Special Features of System-Common Procedures	9-3
Requirements for System-Common Procedures	9-3
Deciding Which Type of Service is Appropriate for Your Task	9-4
Writing System-Common Procedures	9-5
Defining Parameters for a Procedure	9-6
Installation and Deinstallation	9-7
Installation	9-8
Deinstallation	9-11
Defining System-Common Procedure Numbers	9-12
What You Need to Define	9-12
Making a System-Common Label Object File	9-13

Part II - Advanced Topics

10 Stack Format and Calling Conventions	
Introduction	10-1
Memory Addressing	10-1
Program Segmentation	10-2
Memory Organization	10-3
The Medium Model	10-3
Code Segments	10-5
Unallocated Memory	10-5
Notes on the Stack	10-5

Values of DS and SS in Medium Model	10-6
Changing Memory Organization from the Default for Your Compiler	10-7
Code Sharing	10-7
Disposable Initialization Code	10-8
Creating a COED Module	10-8
Disposing of the Code in a COED Module	10-9
Using DS Allocation	10-9
What Is DS Allocation?	10-9
DS Allocation in Real Mode	10-10
DS Allocation in Protected Mode	10-10
Procedure Calls and the Stack	10-11
Overview	10-11
Parameter Passing	10-12
Standard Stack Format	10-12
Standard Prolog and Epilog	10-13
11 Mixed-Language Programming	
Issues in Mixed-Language Programming	11-1
Parameter Passing and the Stack	11-1
Returned Values	11-2
Procedure Initialization and Cleanup	11-3
Calls to CTOS and to the System Libraries	11-3
Model of Computation	11-3
Parameter Naming Convention	11-3
Parameter Passing Convention	11-4
Returned Values	11-4
Mediation	11-5
Calls Between Languages	11-6
Model of Computation	11-7
Parameters and Returned Values	11-7
Run-Time Initialization	11-7
Floating-Point Number Formats	11-7
12 Writing Multi-Process Programs	
Why Use Multiple Processes?	12-1
Process Management	12-2
The Multi-Process Model	12-2
Process States	12-3