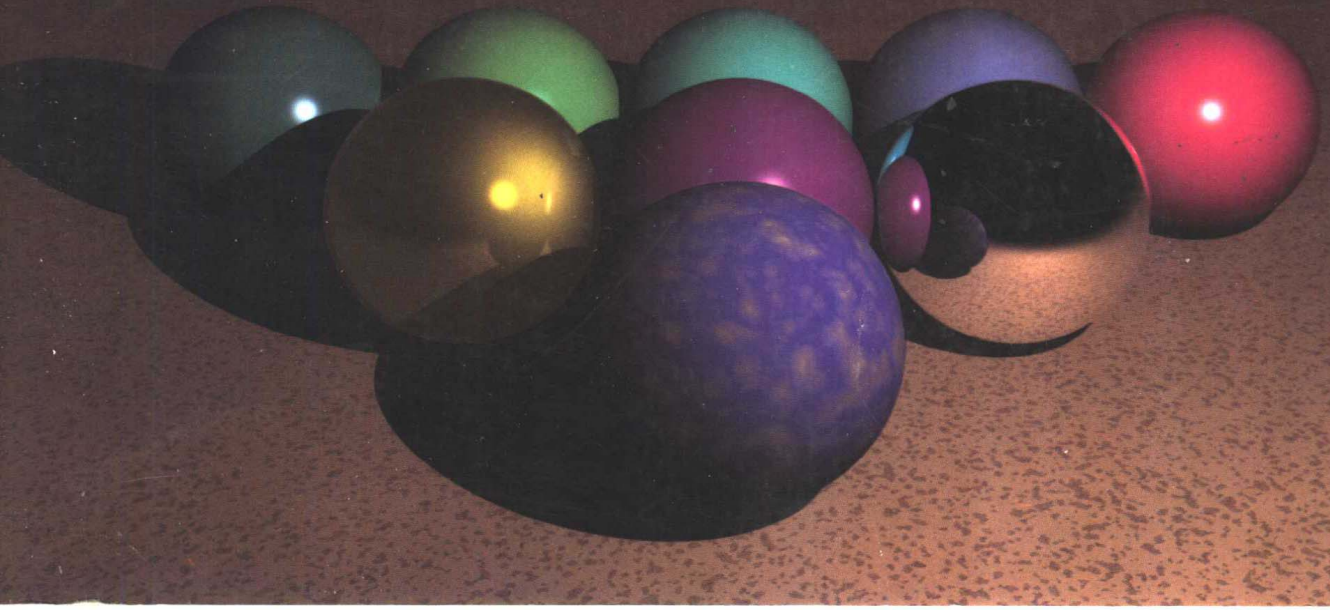
OS/2® Warp Control Program API

Marc Stock



OS/2® Warp Control Program API

Marc Stock

John Wiley & Sons, Inc.

New York • Chichester • Brisbane • Toronto • Singapore

Publisher: Katherine Schowalter
Editor: Theresa Hudson
Managing Editor: Robert S. Aronds
Text Design: Tenenbaum Design
Composition: Impressions, A Division of Edwards Brothers, Inc.

Library of Congress Cataloging-in-Publication Data:

ISBN 0-471-03887-3

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

Dedication

This book is dedicated in loving memory to my grandparents, Dave and Betty Lefkowitz.

Series Introduction

This book is a part of a series of books designed to get programmers the information they need in a quick and efficient manner. Programmers don't have time to hunt through huge unwieldy manuals trying to find the little piece of information they need to finish their tasks. To that end, the design of these books has been optimized by programmers for programmers.

All APIs have been organized into logical groups rather than alphabetically, so, you will find all of the APIs relating to semaphores grouped together in one chapter, for example. The reasoning for this is clear: A programmer looking up how to create a semaphore will most likely want to know how to request it, release it, and so on, immediately afterward.

In addition to grouping related APIs, any information that relates to that group is provided in the same chapter. Each chapter is prefixed with a quick summary of the system architecture related to those APIs, and suffixed with any related structures. In some cases, other related information may be found in the back of the chapter as well.

The architecture summaries at the beginning of the chapters are designed to provide the programmer a quick overview or, in some cases, a refresher on the topic in just a few minutes. I recommend that programmers take the time to read these overviews before delving into design and coding since so many problems arise from poor initial design resulting from a lack of knowledge of the architecture of the topic.

The API pages were designed primarily for efficiency. Therefore, each book in this series may have some small differences on the pages containing the actual APIs. For example, the Workplace Shell reference does not contain any references to what "INCL_" to define for a given API since there is only one for the entire set of Workplace Shell APIs.

It is my sincere hope that you will find the books in this series the quickest, most informative, best organized programmers' reference books you have ever used.

Marc Stock

Preface

This book, like the others in the series, is designed for speedy access to thorough information. There are, however, a couple of things missing from what you would expect in a control program reference. In short, there is no coverage of DosDebug or DosDevIOCtl and their functions. I did not cover these APIs because I felt they didn't fit the scope of this book, which encompasses 95 percent of OS/2 programmers. I also didn't want this book to turn into a huge mega-reference in which you can never find what you are looking for because the book is too unwieldy and vast. For coverage of DosDebug and DosDevIOCtl, I recommend you use the *IBM OS/2 2.0 Technical Library—Control Program Reference.* This book has one other shortfall—no examples. I wanted to write examples that were useful, and used the APIs in context together. This required a lot of additional work that I could not complete in time for publication. Instead, I will make examples available through the OS2DF1 Base APIs section library on CompuServe as soon as I complete them.

I spent more time researching the subjects that people tend to have the most trouble with—like named pipes and EAs. There are numerous behaviors to both of these topics that are either incorrectly documented or not documented at all (as of the time of this writing). I also spent a great deal of time determining system limits, such as how many timers can be started, how many queues, and so forth. I obtained the information via extensive testing and from information I obtained from others. As my testing progressed, I found many pieces of information that neither I, nor most other people, were aware of. And while the book is done, I feel I could go on forever finding more hidden gems in the Control Program APIs. For now, I'll just have to leave them to a second edition. . . .

Marc Stock

Acknowledgments

I would like to thank all the folks on IBM's OS2DF1 forum on CompuServe, who helped me to understand which questions were not being addressed by the currently available material. The forum also proved valuable in obtaining information that is not documented. I highly recommend that programmers sign on to this forum when they get stuck on a piece of code.

I would also like to thank several people who added immeasurably to the design and content of this manuscript: Mindy Pollack for lots of good ideas and comments; Sam Detweiler for putting up with many difficult questions; Matt "Named Pipe" Osborn (need I say more?); and Chris Nack, Greg Cook, and Darian Polliachik for their excellent comments on the architecture summaries.

Finally, I would like to thank my wife, Mary, for her patience and support during this process.

How to Use This Book

This book is designed as a combination guide and reference, with the emphasis on reference. If you are not familiar with the architecture of a particular topic, I recommend that you read the architecture summaries found at the beginning of every chapter. They are short and to the point, but they can be very useful for improving your understanding of the "big picture" or as a quick refresher on an operating system feature you used long ago. The summaries also include a section that discusses related system limitations, warnings, and the list of APIs belonging to that section.

If you are familiar with a particular topic and just need to go to a specific API, the APIs can be found either by looking them up alphabetically in the chapter to which they belong, or from the API index provided in the back. If you don't know to which topic the API belongs, go to the index.

Each API includes a prototype of the call. The data types are in boldface to help guide the eye when trying to determine which types need to be passed. In the parameters section of each call, the parameter name is shown in boldface/italics to aid in quick location of the parameter you are looking for. Note: The prototypes shown in this book are for the C interfaces that are included with the OS/2 Toolkit. If you are using C++, you will find that a few of the parameter types are a little different from the ones shown in this reference. The Toolkit headers for C++ use a new type called PCSZ for pointers to null-terminated constants.

Each API also includes an Other Info section which contains miscellaneous information needed for doing everything from determining which #define is required to include the API's prototype, to which DLL contains the code for that API. Here is an example of an Other Info section:

OTHER INFO

Include file:	bsedos.h	Define:	INCL_DOSFILEMGR
Ordinal:	257	DLL:	DOSCALLS

The Include listed contains the function prototype for the call and any related #defines, structures, and more. Do *not* include this file at the beginning of you code. This information is provided purely for reference purposes. You will also find the name of the #define that should be included to cause the C preprocessor to pull in the necessary function prototypes and #defines so that the compiler will not complain.

Each API also includes a See Also section that includes the names of related APIs that might be of interest. This section includes not only APIs found in this book, but APIs that are a part of Presentation Manager (PM) or the C standard libraries. The calls that are not in this book are shown in italics. For example:

SEE ALSO

DosCreateNPipe -235, DosDupHandle -221, DosOpen -223, DosResetBuffer -247, *fclose*

Following the See Also section, the Notes section contains any related information that may be important to the use of the API. This may be anything from additional details to side-effects. If there is a particular side-effect to an API that is possibly destructive or difficult to detect, then there will also be a Warnings section found after the Notes section. I recommend that the warnings be read every time, even if you intended to look up only the function prototype.

And finally, the Data Structures section found in the back of each chapter contains the definitions for each structure used with the corresponding topic's APIs. The numbers on the right indicate the offset, in bytes, from the beginning of the structure with the total size, in bytes, at the top of the structure. At the end of each structure definition there is a list itemizing which APIs use the structure so that structures can be cross-referenced from API to structure, and from structure to API.

Table of Contents

1

Device Input/Output

In OS/2, communication with devices can be performed with both high-level and low-level interfaces. The high-level interfaces are APIs like DosOpen, DosRead, and DosWrite. These allow the programmer to perform input/output (I/O) on a device with the familiar file system stream interface. Typically, the high-level interfaces are sufficient for most device I/O. The low-level interface for communicating with a device through a device driver is DosDevIOCtl. Additionally, applications can determine which devices are attached by calling DosDevConfig; DosPhysicalDisk will return information on partitionable disks, and DosBeep will generate sound on the computer speaker.

I/O Control Interface

DosDevIOCtl is an expandable I/O control facility that sends commands and data to and from device drivers. The kernel takes generic I/O control packets and reformats them into request packets, and then calls the device driver. The device driver then performs the re-

quested action. DosDevIOCtl can be used with either character or block devices, but before it can be used, a device handle must be obtained by calling DosOpen.

The DosDevIOCtl API is broken up into numerous categories. Each category represents a different device. For example, IOCTL_ASYNC represents the COM port. Within each category are several functions that may be performed specific to that category, like setting the COM port baud rate. The various categories and functions of DosDevIOCtl are documented in IBM's OS/2 2.0 Control Program Reference.

Functions

● DosBeep Device I/O

Generates a specified frequency on the computer speaker.

SYNTAX

APIRET DosBeep(**ULONG** *ulFrequency*, **ULONG** *ulDuration*)

PARAMETERS

ulFrequency - input
The frequency, in Hertz (37 - 32767).
ulDuration - input
The duration of the sound, in milliseconds.

RETURNS

0 NO_ERROR 395 ERROR_INVALID_FREQUENCY

OTHER INFO

Include file: bsedos.h
Ordinal: 286
Define: DOS_PROCESS
DLL: DOSCALLS

SEE ALSO

DosDevIOCtl -4

NOTES

None

● DosDevConfig Device I/O

Returns information about attached devices.

SYNTAX

APIRET DosDevConfig(**PVOID** *pDeviceInfo*, **ULONG** *ulDevType*)

PARAMETERS

pDeviceInfo - output
The address of a buffer that will hold the returned information. Currently, all information returned requires 1 byte.
ulDevType - input
The type of information to return. Specify one of the following:

Constant		Description
DEVINFO_PRINTER	0	The number of attached printers.
DEVINFO_RS232	1	The number of RS232 ports.
DEVINFO_FLOPPY	2	The number of diskette drives.
DEVINFO_COPROCESSOR	3	The presence of a math coprocessor hardware. 0 = no coprocessor, 1 = coprocessor exists.
DEVINFO_SUBMODEL	4	The system submodel byte.
DEVINFO_MODEL	5	The system model byte.
DEVINFO_ADAPTER	6	The type of primary display adapter. 0 = monochrome, 1 = other.