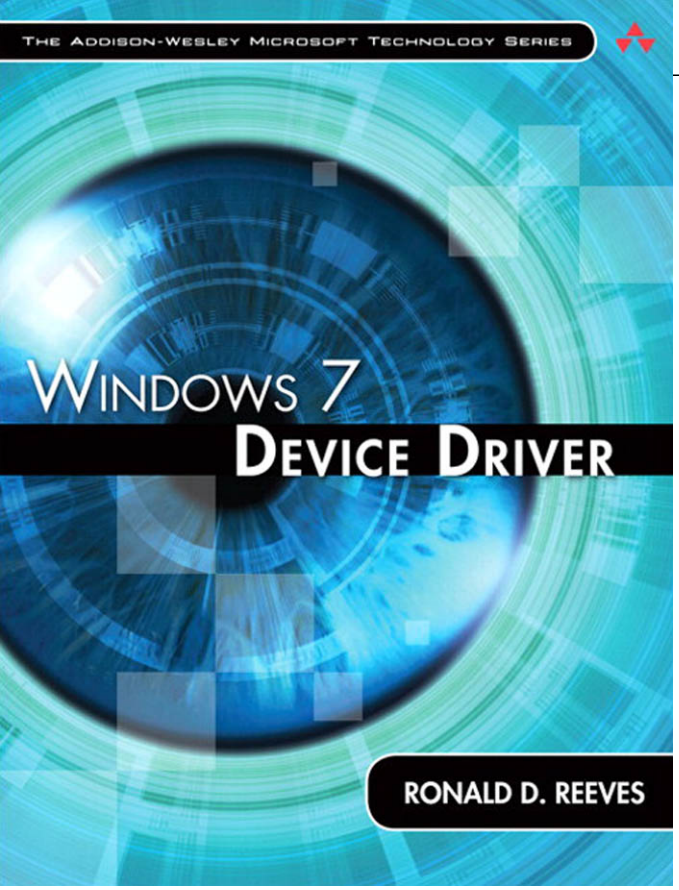
THE ADDISON-WESLEY MICROSOFT TECHNOLOGY SERIES
WINDOWS 7
DEVICE DRIVER
RONALD D. REEVES

Windows 7 Device Driver

Windows 7 Device Driver

Ronald D. Reeves, Ph.D.

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Library of Congress Cataloging-in-Publication Data

Reeves, Ron.
Windows 7 device driver / Ronald D. Reeves.
p. cm.
Includes bibliographical references and index.
ISBN-13: 978-0-321-67021-2 (pbk. : alk. paper)
ISBN-10: 0-321-67021-3 (pbk. : alk. paper)
1. Microsoft Windows device drivers (Computer programs)
I. Title.
QA76.76.D49R44 2011
005.7'1—dc22

2010039109

ISBN-13: 978-0-321-67021-2
ISBN-10: 0-321-67021-3

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.
First printing, November 2010

I would like to dedicate this book to my best friend, and partner in life, my wife, Paulette. Her untiring support and love over the years have been a great source of inspiration.

This page intentionally left blank

CONTENTS

PREFACE

This book provides the technical guidance and understanding needed to write device drivers for the new Windows 7 Operating System. It takes this very complex programming development, and shows how the Windows Driver Framework has greatly simplified this undertaking. It explains the hardware and software architecture you must understand as a driver developer. However, it focuses this around the actual development steps one must take to develop one or the other of the two types of drivers. Thus, this book's approach is a very pragmatic one in that it explains the various software APIs and computer and device hardware based upon our actual device handler development.

There has been great progress in the art of creating and debugging device drivers. There is now a great deal of object-oriented design techniques associated with the driver frameworks that are available to the device driver developer. Much of the previous grunt work, thank goodness, is now being handled by the latest device development framework Windows Driver Foundation (WDF). We will be covering both the user mode and kernel mode of device driver development. WDF has excellent submodels contained within it, called the User Mode Driver Framework and the Kernel Mode Driver Framework.

It is really great to see a Windows Driver Framework involved in the creation of Windows Device Drivers. I started working with Windows in 1990 and we primarily used the Win32 System APIs to communicate and control the Windows Operating System for our applications. We used the Device Driver Kit (DDK) to create the Windows drivers. Because I had my own company to create application software, I obviously was very concerned about the time it took to develop application software, and the robustness of the application. There were more than 2,000 Win32 APIs to be used for this task.

Then in about 1992, Microsoft came out with the Microsoft Framework Classes (MFC). In these 600+ classes, most of the Win32 APIs were encapsulated. Of course, prior to this, around 1988, the C++ compiler came out, and Object Oriented Programming started to come

into its own. By using the MFC Framework, we could produce more application software faster and with better quality. My return on investment (ROI) went up, and I made more money. This sure made a believer of me in the use of frameworks. I used MFC until the .NET Framework came out, and for the last nine years I have been using this great collection of classes. All along, Microsoft was working to bring this same kind of software development improvements to developing device drivers. We came from the DDK, to the Windows Driver Model, to the Windows Driver Foundation Framework.

Therefore, this book shows how to create Windows 7 Device Drivers using the Windows Driver Foundation Framework. This should give us driver developers a little more sanity when meeting our deadlines.

The book is broken into three major parts as follows:

- **Part I, "Device Driver Architecture Overview"**—This part lays out the architecture involved in both software and hardware for device handler development. It also covers the driver development environment needed for driver development, for both types of drivers that are normally developed—that is, User Mode and Drivers. This section also covers the two Windows driver frameworks that are most commonly used for driver device development today, which are part of the Windows Driver Framework (WDF). These two Windows Driver Frameworks are the User Mode Driver Framework (UMDF) and the Kernel Mode Driver Framework (KMDF).

- **Part II, "User Mode Drivers"**—This part outlines the approach, design, development, and debug of User Mode Drivers. This part takes the driver programmer from start to finish in developing User Mode Drivers. We primarily use the User Mode Driver Framework for all of this work. The code is done in C++ because it is the best way to develop these types of drivers. Discussions are based on a USB User Mode Driver that we will develop using the UMDF. We will use a USB hardware learning kit from Open Systems Resources, Inc. (OSR). This provides a hardware simulation to test our User Mode Drivers. This part is primarily stand-alone and could be read and used without reading any other parts of the book. However, you will probably want to read Part I to get a feel for what we are using.

- **Part III, "Kernel Mode Drivers"**—This part outlines the approach, design, development, and debug of Kernel Mode Drivers. The intent again is to take the driver programmer from start to finish in developing Kernel Mode Drivers. For this section, we primarily use the Kernel Mode Driver Framework for all of this work. The code is done in C because this is the best way to develop these types of drivers. Discussions are based on a Kernel Mode Driver that we develop using the KMDF. We use a Peripheral Component Interconnect (PCI) hardware learning kit from OSR. This provides a hardware simulation to test our Kernel Mode Drivers. The section is also primarily stand-alone and could be read and used without reading any other parts of the book. Again, you will probably want to read Part I to get a feel for what we are using.

ACKNOWLEDGMENTS

I am most grateful to my editor Bernard Goodwin at Pearson Education for giving me the opportunity to write this book. His support during the preparation was great. I would also like to thank his assistant Michelle Housley for her timely fashion in getting me reference books and material. Also, I would like to thank John Herrin, Video Project Manager at Pearson Education, for support and help in creating the book video. Thanks to Michael Thurston, my development editor, for making the book sound very polished.

This page intentionally left blank

About the Author

Ronald D. Reeves, Ph.D., is founder and president of Software Genesis, LLC, a software development and consulting company based in Brighton, Michigan. Dr. Reeves has some forty years of experience in designing and developing computer hardware and software applications. He holds degrees in engineering and computer science and is a nationally recognized author, consultant, and teacher.

If you have questions, comments, or suggestions for improving this book, we would like to hear from you. You can contact the author by U.S. Mail or by email at the following addresses:

Dr. Ronald D. Reeves
PO Box 2425
Brighton, MI 48116
Email: software.genesis@att.net

This page intentionally left blank

INTRODUCTION

Device drivers are where the rubber meets the road, and are very specialized pieces of software that allow your application programs to communicate to the outside world. Any communications your Windows 7 makes to the outside world requires a Device Driver. These devices include such things as mouse, display, keyboard, CD-ROMS, data acquisition, data network communication, and printers. However, Microsoft has written and supplied a great many drivers with the Windows 7 Operating System. These drivers support most of what we call the standard devices, and we will not be covering them in this book.

This book is about how we create device drivers for the nonstandard devices—devices that are not typically found on standard PCs. Quite often, the market is too small for Microsoft to create a standard device driver for these types of devices—such things as data acquisition boards, laboratory equipment, special test equipment, and communications boards.

This discussion will highlight the significant features of interest to the device driver developers. Figure I.1 shows a general block diagram of Windows 7. We develop more detailed block diagrams in the discussions in various parts of the book.

In Figure I.1 the user applications don't call the Windows 7 Operating System Services directly. They go thru the Win32 subsystem dynamic-linked libraries (DLL). The User Mode Device Drivers, discussed later, go through this same communication channel.

The various Windows 7 services that run independently are handled by the Service Processes. They are typically started by the service control manager.

The various Windows 7 System Support Processes are not considered Windows 7 services. They are therefore not started by the service control manager.

The Windows 7 I/O Manager actually consists of several executive subsystems that manage hardware devices, priority interfaces for both the system and the applications. We cover this in detail in Parts II and III of this book.

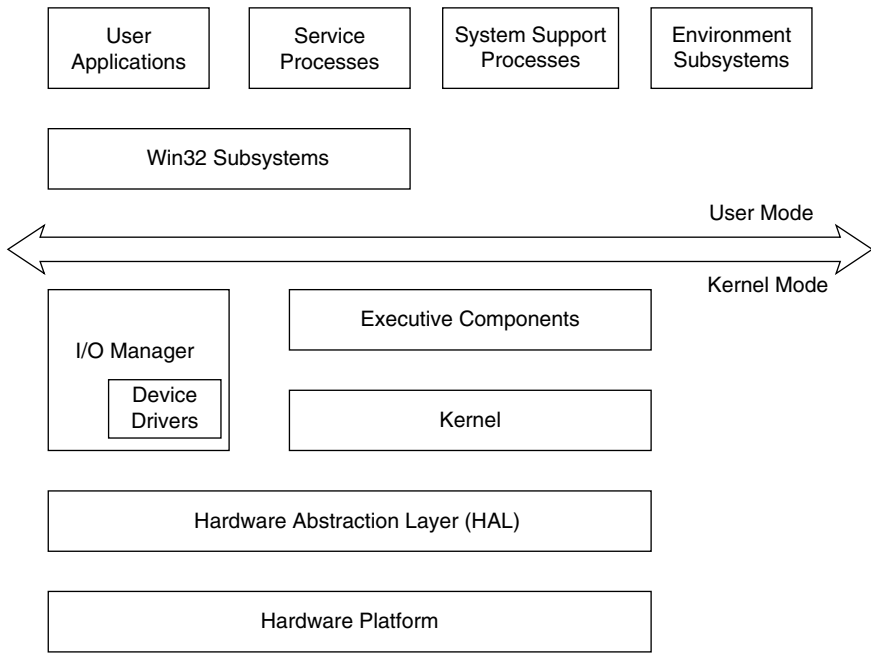

Figure I.1 System Overview Windows 7

The Device Driver block shown in the I/O Manager block is primarily what this book is all about—that is, designing, developing, and testing Windows 7 Device Drivers. The drivers of course translate user I/O function calls into hardware device I/O requests.

The Hardware Abstraction Layer (HAL) is a layer of code that isolates platform-specific hardware differences from the Windows 7 Operating System. This allows the Windows 7 Operating System to run on different hardware motherboards. When device driver code is ported to a new platform, in general, only a recompile is necessary. The device driver code relies on code (macros) within HAL to reference hardware buses and registers. HAL usage in general is implemented such that inline performance is achieved.

The Windows 7 performance goals often impact device driver writers. When system threads and users request service from a device, it's very important that the driver code not block execution. In this case, where the driver request cannot be handled immediately, the request must be

queued for subsequent handling. As we will show in later discussions, the I/O Manager routines available allow us to do this.

Windows 7 gives us a rich architecture for applications to utilize. However, this richness has a price that device driver authors often have to pay. Microsoft, realizing this early on some 14 years ago, started developing the driver development models and framework to aid the device driver author. The earliest model, the Windows Driver Model (WDM) had a steep learning curve, but was a good step forward. Microsoft has subsequently developed the Windows Driver Foundation (WDF) that makes developing robust Windows 7 drivers easier to implement and learn. This book is about developing Windows 7 Device Driver using WDF.

This page intentionally left blank