



The Windows® 2000 Device Driver Book

A GUIDE FOR PROGRAMMERS

Second Edition

- The #1 Windows device driver book—fully updated for Windows 2000!
- Step-by-step planning, implementation, testing, debugging, installation, and distribution
- Complete coverage of the new Windows Driver Model (WDM)
- Practical debugging and interactive troubleshooting
- CD-ROM: Exclusive tools for streamlining driver development, plus extensive C/C++ sample driver library!

Art Baker
Jerry Lozano

Foreword by Andrew Scappu

UCI CORPORATION



MICROSOFT® TECHNOLOGIES SERIES

Foreword xxv

Preface xxvii

▼ ONE Introduction to Windows 2000 Drivers 1

Overall System Architecture 1

Design Goals for Windows 2000 1

Hardware Privilege Levels in Windows 2000 2

Portability 3

Extendibility 3

Performance 4

Executive Components 5

System Service Interface 5

Object Manager 5

Configuration Manager 5

Process Manager 5

Virtual Memory Manager 6

Local Procedure Call Facility 6

I/O Manager 7

Active Directory Service 7

Extensions to the Base Operating System 7

The Win32 Subsystem 8

Integral Subsystems 9

Kernel-Mode I/O Components 10

Design Goals for the I/O Subsystem 10

Kinds of Drivers in Windows 2000 11

Special Driver Architectures 13

Video Drivers 14

Printer Drivers 15

Multimedia Drivers 17

Network Drivers 17

Summary 18

▼ TWO The Hardware Environment 20

Hardware Basics 20

Device Registers 21

Command 21

Status 21

Data 21

Accessing Device Registers	21
<i>I/O Space Registers</i>	22
<i>Memory-Mapped Registers</i>	23
Device Interrupts	24
<i>Interrupt Priority</i>	25
<i>Interrupt Vectors</i>	25
<i>Signaling Mechanisms</i>	25
<i>Processor Affinity</i>	26
Data Transfer Mechanisms	26
<i>Programmed I/O</i>	27
<i>Direct Memory Access</i>	27
DMA Mechanisms	28
<i>System DMA</i>	28
<i>Bus Master DMA</i>	28
Device-Dedicated Memory	28
Auto-regulation and Auto-configuration	29
<i>Device Resource Lists</i>	29
<i>No Jumpers or Switches</i>	29
<i>Change Notification</i>	29
Buses and Windows 2000	30
ISA:The Industry Standard Architecture	30
<i>Register Access</i>	31
<i>Interrupt Mechanisms</i>	31
<i>DMA Capabilities</i>	31
<i>Automatic Recognition and Configuration</i>	33
EISA:The Extended Industry Standard Architecture	33
<i>Register Access</i>	33
<i>Interrupt Mechanisms</i>	33
<i>DMA Capabilities</i>	33
<i>Device Memory</i>	35
<i>Automatic Recognition and Configuration</i>	35
PCI:The Peripheral Component Interconnec	35
<i>Register Access</i>	37
<i>Interrupt Mechanisms</i>	37
<i>DMA Capabilities</i>	37
<i>Device Memory</i>	38
<i>Autoconfiguration</i>	38
USB:The Universal Serial Bus	39
<i>Register Access</i>	39
<i>Interrupt Mechanisms</i>	40
<i>DMA Capabilities</i>	41
<i>Automatic Recognition and Configuration</i>	41
IEEE 1394:The Firewire™ Bus	41
<i>Register Access</i>	41
<i>Interrupt Mechanisms</i>	41
<i>DMA Capabilities</i>	41
<i>Automatic Recognition and Configuration</i>	41

The PC Card (PCMCIA) Bus 41

Register Access 42

Interrupt Mechanisms 42

DMA Capabilities 42

Automatic Recognition and Configuration 42

Hints for Working with Hardware 44

Learn about the Hardware 44

Bus Architecture 44

Control Registers 44

Error and Status Reporting 44

Interrupt Behavior 45

Data Transfer Mechanisms 45

Device Memory 45

Make Use of Hardware Intelligence 45

Test the Hardware 45

Basic Tests 45

Standalone Tests 46

Summary 46

▼ THREE Kernel-Mode I/O Processing 47

How Kernel-Mode Code Executes 47

Trap or Exception Context 48

Interrupt Context 48

Kernel-Mode Thread Context 48

Use of Interrupt Priorities by Windows 2000 49

CPU Priority Levels 49

Interrupt Processing Sequence 50

Software-Generated Interrupts 50

Deferred Procedure Calls (DPCs) 51

Operation of a DPC 51

Behavior of DPCs 52

Accessing to User Buffers 53

Buffer Access Mechanisms 53

Structure of a Kernel-Mode Driver 54

Driver Initialization and Cleanup Routines 54

DriverEntry Routine 54

Reinitialization Routine 55

Unload Routine 55

Shutdown Routine 55

Bugcheck Callback Routine 55

I/O System Service Dispatch Routines 55

Open and Close Operations 55

Device Operations 56

Data Transfer Routines 56

Start I/O Routing 56

<i>Interrupt Service Routine (ISR)</i>	56
<i>DPC Routines</i>	56
Resource Synchronization Callbacks	57
<i>ControllerControl Routine</i>	57
<i>AdapterControl Routine</i>	58
<i>SynchCriticalSection</i>	58
Other Driver Routines	58
<i>I/O Processing Sequence</i>	58
Preprocessing by the I/O Manager	59
Preprocessing by the Device Driver	59
Device Start and Interrupt Service	60
<i>Start I/O</i>	60
<i>ISR</i>	60
Postprocessing by the Driver	61
Postprocessing by the I/O Manager	61
Summary	62
 ▼ FOUR Drivers and Kernel-Mode Objects	63
<i>Data Objects and Windows 2000</i>	63
Windows 2000 and OOP	64
Windows 2000 Objects and Win32 Objects	64
<i>I/O Request Packets (IRPs)</i>	64
Layout of an IRP	65
<i>IRP Header</i>	66
<i>I/O Stack Locations</i>	66
Manipulating IRPs	67
<i>IRPs as a Whole</i>	68
<i>IRP Stack Locations</i>	68
<i>Driver Objects</i>	68
Layout of a Driver Object	69
<i>Device Objects and Device Extensions</i>	70
Layout of the Device Object	70
Manipulating Device Objects	71
Device Extensions	72
<i>Controller Objects and Controller Extensions</i>	73
Layout of the Controller Object	73
Manipulating Controller Objects	74
Controller Extensions	74
<i>Adapter Objects</i>	75
Layout of an Adapter Object	76
Manipulating Adapter Objects	77
<i>Interrupt Objects</i>	77
Layout of an Interrupt Object	77

Summary 78

▼ **FIVE General Development Issues** 79

Driver Design Strategies 79

Use Formal Design Methods 79

Use Incremental Development 80

Examine and Use the Sample Drivers 81

Coding Conventions and Techniques 81

General Recommendations 81

Naming Conventions 82

Header Files 83

Status Return Values 83

Windows 2000 Driver Support Routines 84

Discarding Initialization Routines 85

Controlling Driver Paging 86

Driver Memory Allocation 87

Memory Available to Drivers 87

Working with the Kernel Stack 87

Working with the Pool Areas 88

System Support for Memory Suballocation 89

Zone Buffers 89

Lookaside Lists 90

Unicode Strings 91

Unicode String Data Types 91

Working with Unicode 92

Interrupt Synchronization 95

The Problem 95

Interrupt Blocking 96

Rules for Blocking Interrupts 96

Synchronization Using Deferred Procedure Calls 97

Synchronizing Multiple CPUs 97

How Spin Locks Work 98

Using Spin Locks 98

Rules for Using Spin Locks 99

Linked Lists 100

Singly Linked Lists 100

Doubly Linked Lists 101

Removing Blocks from a List 101

Summary 102

▼ SIX Initialization and Cleanup Routines 103

Writing a DriverEntry Routine 103

- Execution Context 104
- What a DriverEntry Routine Does 104
- Announcing DriverEntry Points 105
- Creating Device Objects 105
- Choosing a Buffering Strategy 107
- Device Names 107

Code Example: Driver Initialization 108

- DriverEntry 108
- CreatDevice 110

Writing Reinitialize Routines 111

- Executive Context 111
- What a Reinitialize Routine Does 112

Writing and Unload Routine 112

- Execution Context 112
- What an Unload Routine Does 113

Code Example: Driver Unload 113

Writing Shutdown Routines 114

- Execution Context 115
- What a Shutdown Routine Does 115
- Enabling a Shutdown Notification 115

Testing the Driver 116

- Testing Procedure 116
- Visual C++ Device Driver AppWizard 116
- The Windows 2000 DDK 117
- Results of the Driver Build 117
- Installing a Kernel-Mode Driver Manually 117
- Loading the Driver 118
- Windows 2000 Computer Management Console 118
- The WINOBJ Utility 119

Summary 120

▼ SEVEN Driver Dispatch Routines 122

Announcing Driver Dispatch Routines 122

- I/O Request Dispatching Mechanism 123
- Enabling Specific Function Codes 123
- Deciding Which Function Codes to Support 124

Writing Driver Dispatch Routines 125

- Execution Context 126

What Dispatch Routines Do	127
Exiting the Dispatch Routine	127
<i>Signaling an Error</i>	127
<i>Completing a Request</i>	128
<i>Scheduling a Device Operation</i>	129
Processing Read and Write Requests	129
User Buffer Access	130
<i>Buffered I/O</i>	130
<i>Direct I/O</i>	130
<i>Neither Method</i>	131
Code Example: A Loopback Device	131
Extending the Dispatch Interface	133
Defining Private IOCTL Values	134
IOCTL Argument-Passing Methods	134
Writing IOCTL Header Files	136
Processing IOCTL Requests	136
Managing IOCTL Buffers	138
<i>METHOD_BUFFERED</i>	138
<i>METHOD_IN_DIRECT</i>	138
<i>METHOD_OUT_DIRECT</i>	139
<i>METHOD_NEITHER</i>	139
Testing Driver Dispatch Routines	139
Testing Procedure	139
Sample Test Program	140
Summary	140
 ▼ EIGHT Interrupt-Driven I/O	 141
How Programmed I/O Works	141
What Happens During Programmed I/O	142
Synchronizing Driver Routines	142
Driver Initialization and Cleanup	143
Initializing the Start I/O Entry Point	144
Initializing a DpcForIsr Routine	144
Connecting to an Interrupt Source	144
Disconnecting from an Interrupt Source	146
Writing a Start I/O Routine	146
Execution Context	146
What the Start I/O Routine Does	146
Writing an Interrupt Service Routine (ISR)	147
Execution Context	147
What the Interrupt Service Routine Does	148
Writing a DpcForIsr Routine	149
Execution Context	149

What the DpcForIsr Routine Does	149
Priority Increments	150
<i>Some Hardware: The Parallel Port</i>	150
How the Parallel Port Works	150
Device Registers	151
Interrupt Behavior	152
A Loopback Connector for the Parallel Port	153
<i>Code Example: Parallel Port Loopback Driver</i>	153
Purpose of the Driver	153
Driver.H	154
Driver.cpp	154
CreateDevice	155
DispatchWrite	156
DispatchRead	156
StartIo	156
ISR	157
DpcForIsr	158
<i>Testing the Parallel Port Loopback Driver</i>	159
Testing Procedure	159
<i>Summary</i>	160
 ▼ NINE Hardware Initialization	 161
<i>The Plug and Play Architecture: A Brief History</i>	162
Goals of Plug and Play	162
Components of Plug and Play	163
Plug and Play Manager	163
Power Manager	163
Registry	163
INF Files	163
Plug and Play Drivers	163
<i>The Role of the Registry for Legacy Drivers</i>	164
<i>Detecting Devices with Plug and Play</i>	165
<i>The Role of Driver Layers in Plug and Play</i>	166
<i>The New WDM IRP Dispatch Functions</i>	170
Required Plug and Play IRPs	171
PDO Plug and Play IRPs	172
Passing Down Plug and Play Requests	172
I/O Completion Routines	175
Bus Driver Plug and Play Requests	178
<i>Device Enumeration</i>	178/
Hardware Resource Descriptions	179
Using Hardware Resources within the Driver	181
<i>Driver Interfaces</i>	181
Interface Definition	182

Interface Construction	182
Interface Reference Counting	183
Registering and Enabling an Interface	183
<i>Code Example: A Simple Plug and Play Driver</i>	185
<i>Summary</i>	185

▼ TEN Power Management 186

<i>Hot Plug Devices</i>	187
Bus Considerations	187
Device Considerations	187
<i>OnNow Initiative</i>	188
Power States	188
Power Policies	189
Power State Matrix	190
Power State Changes	190
<i>Wake Requests</i>	194
Canceling the Wake-Armed IRP	196
<i>Power Management Issues</i>	196
Idle Management	197
User Interface for Power Management	198
<i>Summary</i>	198

▼ ELEVEN Timers 199

<i>Handling Device Timeouts</i>	199
How I/O Timer Routines Work	199
How to Catch Device Timeout Conditions	200
<i>Code Example: Catching Device Timeouts</i>	201
Device Extension Additions	201
AddDevice Additions	202
Create Dispatch Routine Changes	202
StartIo Changes	202
ISR Changes	203
I/O Timer Callback Routine	203
<i>Managing Devices without Interrupts</i>	204
Working with Polled Devices	204
How CustomTimerDpc Routines Work	206
How to Set Up a CustomTimerDpc Routine	207
How to Specify Expiration Times	208
Other Uses for CustomTimerDpc Routines	210
<i>Code Example: A Timer-Based Driver</i>	210
Device Extension Additions	210
AddDevice Modifications	211

TransmitBytes Changes	211
PollingTimerDpc Routine	212
Summary	212
▼ TWELVE DMA Drivers	214
<i>How DMA Works under Windows</i>	<i>214</i>
Hiding DMA Hardware Variations with Adapter Objects	215
The Scatter/Gather Problem	216
Memory Descriptor Lists	217
Maintaining Cache Coherency	219
CPU Data Cache	219
Adapter Object Cache	221
Packet-Based and Common Buffer DMA	221
Limitations of the Windows 2000 DMA Architecture	222
<i>Working with Adapter Objects</i>	<i>222</i>
Finding the Right Adapter Object	222
Acquiring and Releasing the Adapter Object	225
Setting Up the DMA Hardware	226
Flushing the Adapter Object Cache	227
<i>Writing a Packet-Based Slave DMA Driver</i>	<i>228</i>
How Packet-Based Slave DMA Works	228
IRP_MN_START_DEVICE Handler	228
Start I/O Routine	229
Adapter Control Routine	229
Interrupt Service Routine	230
DpcForIsr Routine	230
Splitting DMA Transfers	231
First Transfer	231
Additional Transfers	232
<i>Code Example: A Packet-Based Slave DMA Driver</i>	<i>233</i>
DRIVER.H	233
GetDmaInfo Routine	234
Start I/O Changes	235
AdapterControl Routine	236
DpcForIsr Routine	237
<i>Writing a Packet-Based Bus Master DMA Driver</i>	<i>238</i>
Setting Up Bus Master Hardware	239
AdapterControl Routine	239
DpcForIsr Routine	241
Hardware with Scatter/Gather Support	242
Building Scatter/Gather Lists with Map Transfer	243
AdapterControl Routine	243
DpcForIsr Routine	244
<i>Writing a Common Buffer Slave DMA Driver</i>	<i>245</i>
Allocating a Common Buffer	245

Using Common Buffer Slave DMA to Maintain Throughput 246

AddDevice Routine 247

IRP_MN_START_DEVICE Handler 247

Dispatch Routine 248

Start I/O Routine 249

Interrupt Service Routine 249

DpcForIsr Routine 249

IRP_MN_STOP_DEVICE Handler 249

Writing Common Buffer Bus Master DMA Driver 250

How Common-Buffer Bus Master DMA Works 250

IRP_MN_START_DEVICE Handler 250

Start I/O Routine 251

Interrupt Service Routine 251

IRP_MN_STOP_DEVICE Handler 251

Summary 251

▼ THIRTEEN Windows Management and Instrumentation 253

WMI: The Industry Picture 254

The WMI Architecture 255

Providing WMI Support in a WDM Driver 256

MOF Syntax 257

Example MOF Class Definition 258

Compiling the MOF Source 260

Handling WMI IRP Requests 261

Classes and Instances 262

WMILIB 263

DpWmiQueryReginfo 265

DpWmiQueryDataBlock 266

DpWmiSetDataBlock 266

DpWmiSetDataItem 266

DpWmiExecuteMethod 268

DpWmiFunctionControl 268

WMI Summary 270

Conventional Driver Event Logging 270

How Event Logging Works 270

Working with Messages 271

Writing a Message Definition File 272

A Simple Example 273

Header Section 274

Message Section 274

Compiling a Message Definition File 275

Adding Message Resources to a Driver 276

Registering a Driver as an Event Source 276

Generating Log Entries 276

Allocating an Error-Log Packet 276

Logging the Error 278

Summary 279

▼ FOURTEEN System Threads 280

Definition and Use of System Threads 280

When to Use Threads 281

Creating and Terminating System Threads 282

Managing Thread Priority 282

System Worker Threads 283

Thread Synchronization 283

Time Synchronization 284

General Synchronization 284

KeWaitForSingleObject 284

KeWaitForMultipleObjects 284

Using Dispatcher Objects 286

Event Objects 286

Sharing Events Between Drivers 288

Mutex Objects 288

Semaphore Objects 289

Timer Objects 290

Thread Objects 292

Variations on the Mutex 293

Fast Mutexes 293

Executive Resources 293

Synchronization Deadlocks 294

Code Example: A Thread-Based Driver 295

How the Driver Works 295

The `DEVICE_EXTENSION` Structure 296

The `AddDevice` Function 297

The `DispatchReadWrite` Function 298

`Thread.C` 299

WorkerThreadMain 299

KillThread 300

`Transfer.C` 301

PerformDataTransfer 301

AcquireAdapterObject and AdapterControl 304

PerformSynchronousTransfer 305

DpcForIsr 306

Summary 307

▼ FIFTEEN Layered Drivers 308

An Overview of Intermediate Drivers 309

Intermediate Drivers Defined 309

When to Use a Layered Architecture 310

Pros of Layered Architecture 310

Cons of Layered Architecture 310

Writing Layered Drivers 311

How Layered Drivers Work 311

Initialization and Cleanup in Layered Drivers 311

DriverEntry 312

AddDevice 312

RemoveDevice 313

Code Fragment: Connecting to Another Driver 313

Other Initialization Concerns for Layered Drivers 314

Transparent 314

Virtual or Logical Device Layer 315

I/O Request Processing in Layered Drivers 315

Complete the Original IRP 315

Pass the IRP to Another Driver 316

Allocate Additional IRPs 317

Code Fragment: Calling a Lower-Level Driver 317

Writing I/O Completion Routines 318

Requesting an I/O Completion Callback 318

Execution Context 319

What I/O Completion Routines Do 320

Release the Original IRP 320

Deallocate the IRP 320

Recycle the IRP 320

Code Fragment: An I/O Completion Routine 231

Allocating Additional IRPs 322

The IRP's I/O Stack Revisited 323

Controlling the Size of the IRP Stack 324

Creating IRPs with IoBuildSynchronousFsdRequest 325

Creating IRPs with IoBuildAsynchronousFsdRequest 327

Creating IRPs with IoBuildDevice IoControlRequest 328

Creating IRPs from Scratch 329

IRPs from IoAllocateIrp 329

IRPs from ExAllocatePool 331

IRPs from Driver-Managed Memory 332

Setting Up Buffers for Lower Drivers 332

Buffered I/O Requests 332

Direct I/P Requests 333

Keeping Track of Driver-Allocated IRPs 333

Synchronous I/O 333

Asynchronous I/O 334

Writing Filter Drivers 335

How Filter Drivers Work 335

Initialization and Cleanup in Filter Drivers 336

AddDevice Routine 336

RemoveDevice Routine 337

Making the Attachment Transparent 337

Code Example: A Filter Driver 338

The DEVICE_EXTENSION Structure 338

The DriverEntry Function	338
The AddDevice Function	339
<i>GetBufferLimits</i>	340
The OverriddenDispatchWrite Function	341
The OverriddenDispatchDeviceIoControl Function	342
The DispatchPassThru Function	343
The I/O Completion Routines	343
<i>WriteCompletion</i>	344
<i>GenericCompletion</i>	345

Writing Tightly Coupled Drivers 346

How Tightly Coupled Drivers Work	346
Initialization and Cleanup in Tightly Coupled Drivers	346
<i>Lower AddDevice Routine</i>	346
<i>Upper AddDevice Routine</i>	347
<i>Upper RemoveDevice Routine</i>	348
<i>Lower RemoveDevice Routine</i>	348

Summary 349

▼ SIXTEEN Driver Installation 349

Installation of a Driver 349

Auto-Install Using INF Files 350

INF File Structure	350
Version Section	351
Manufacturers Section	351
Models Section	351
DDInstall Section	352
CopyFiles Section	353
AddReg Section	354
SourceDisksNames Section	356
SourceDisksFiles Section	357
<i>DestinationDirs</i> Section	357
<i>DDInstall.Services</i> Section	357
<i>ServiceInstall</i> Section	359
<i>INF Example</i>	359
<i>Validating INF Syntax</i>	361

Using a Driver INF File 361

Manual Installation	362
Automatic Installation	362
The Add/Remove Hardware Wizard	362
Class Names and Device IDs	36
Customizing an Installation	364

Controlling Driver Load Sequence 364

Driver Stack Order	367
--------------------	-----

<i>Digital Signing of a Driver</i>	367
Why Microsoft Verifies Drivers	368
Digital Signatures	368
<i>Summary</i>	369

▼ SEVENTEEN Testing and Debugging Drivers 370

Guidelines for Driver Testing 371

A Generalized Approach to Testing Drivers 371

When to Test 371

What to Test 372

How to Develop the Tests 372

How to Perform the Tests 372

Who Should Perform the Tests 373

The Microsoft Hardware Compatibility Tests 373

Why Drivers Fail 374

Categories of Driver Errors 374

Hardware Problems 374

System Crashes 374

Resource Leaks 375

Thread Hangs 375

System Hangs 376

Reproducing Driver Errors 376

Time Dependencies 376

Multiprocessor Dependencies 377

Multithreading Dependencies 377

Other Causes 377

Defensive Coding Strategies 377

Keeping Track of Driver Bugs 378

Reading Crash Screens 378

What Happens When the System Crashes 378

The Blue Screen of Death 379

An Overview of WinDbg 380

The Key to Source Code Debugging 381

Symbol Directories 381

Source Code Directories 381

Some WinDbg Commands 381

Analyzing a Crash Dump 383

Goals of the Analysis 383

Starting the Analysis 383

Tracing the Stack	385
<i>High IRQL Crashes</i>	385
<i>Crashes Below DISPATCH_LEVEL</i>	385
Indirect Methods of Investigation	386
<i>Finding I/O Requests</i>	386
<i>Examining Processes</i>	387
Interactive Debugging	389
Starting and Stopping a Debug Session	389
Setting Breakpoints	391
Setting Hard Breakpoints	391
Intermediate Output	392
Writing WinDbg Extensions	392
How WinDbg Extensions Work	392
Initialization and Version-Checking Functions	393
<i>WinDbgExtensionDllInit</i>	393
<i>ExtensionApiVersion</i>	393
<i>CheckVersion</i>	393
Writing Extension Commands	393
WinDbg Helper Functions	394
Building and Using an Extension DLL	396
Code Example: A WinDbg Extension	396
DBG.C	396
<i>Header</i>	396
<i>Globals</i>	396
<i>Required Functions</i>	398
<i>Command Routines</i>	399
<i>Sample Output</i>	400
Miscellaneous Debugging Techniques	400
Leaving Debugged Code in the Driver	401
Catching Incorrect Assumptions	401
Using Bugcheck Callbacks	402
Catching Memory Leaks	403
Using Counters, Bits, and Buffers	404
<i>Sanity Counters</i>	404
<i>Event Bits</i>	404
<i>Trace Buffers</i>	404
Summary	406
A. The Driver Debug Environment	407
B. Bugcheck Codes	414
C. Building Drivers	431
Bibliography	439
Index	441