

42-266-4000

The Windows NT Device Driver Book: A Guide for Programmers

Art Baker

Cydonix Corporation



To join a Prentice Hall PTR Internet mailing list, point to:
<http://www.prenhall.com/register>



Prentice Hall PTR
Upper Saddle River, New Jersey 07458
<http://www.prenhall.com>

Contents

PREFACE.....	xv
ACKNOWLEDGMENTS	xx
CHAPTER 1 INTRODUCTION TO WINDOWS NT DRIVERS.....	1
1.1 OVERALL SYSTEM ARCHITECTURE	1
Design Goals for Windows NT.....	1
Hardware Privilege Levels in Windows NT	2
Base Operating System Components	2
What's in the Executive	4
Extensions to the Base Operating System.....	7
More about the Win32 Subsystem	8
1.2 KERNEL-MODE I/O COMPONENTS	10
Design Goals for the I/O Subsystem.....	10
Layered Drivers in Windows NT.....	10
SCSI Drivers	12
Network Drivers.....	13
1.3 SPECIAL DRIVER ARCHITECTURES	15
Video Drivers	15
Printer Drivers.....	17
Multimedia Drivers	20
Drivers for Legacy 16-bit Applications	21
1.4 SUMMARY.....	23

CHAPTER 2 THE HARDWARE ENVIRONMENT	24
2.1 HARDWARE BASICS.....	24
Device Registers.....	25
Accessing Device Registers	26
Device Interrupts	27
Data Transfer Mechanisms	29
Direct Memory Access (DMA) Mechanisms.....	30
Device-Dedicated Memory	31
Requirements for Autoconfiguration	32
2.2 BUSES AND WINDOWS NT	33
ISA—The Industry Standard Architecture.....	33
MCA—The Micro Channel Architecture	36
EISA—The Extended Industry Standard Architecture	39
PCI—The Peripheral Component Interconnect	41
2.3 HINTS FOR WORKING WITH HARDWARE.....	45
Learn about the Hardware	45
Make Use of Hardware Intelligence.....	46
Test the Hardware	46
2.4 SUMMARY.....	47
CHAPTER 3 KERNEL-MODE I/O PROCESSING	48
3.1 HOW KERNEL-MODE CODE EXECUTES.....	48
Exceptions	48
Interrupts	49
Kernel-Mode Threads	49
3.2 USE OF INTERRUPTS BY NT	49
CPU Priority Levels	49
Interrupt Processing Sequence	50
Software-Generated Interrupts	51
3.3 DEFERRED PROCEDURE CALLS (DPCs)	51
Operation of a DPC	51
Behavior of DPCs	52
3.4 ACCESS TO USER BUFFERS.....	53
Buffer-Access Mechanisms	53
3.5 STRUCTURE OF A KERNEL-MODE DRIVER.....	54
Driver Initialization and Cleanup Routines.....	55
I/O System Service Dispatch Routines	55
Data Transfer Routines	56
Resource Synchronization Callbacks	57
Other Driver Routines	57
3.6 I/O PROCESSING SEQUENCE	58
Request Preprocessing by NT	58
Request Preprocessing by the Driver	59
Data Transfer	59
Postprocessing by the Driver	60
Postprocessing by the I/O Manager	60
3.7 SUMMARY.....	61

CHAPTER 4 DRIVERS AND KERNEL-MODE OBJECTS	62
4.1 DATA OBJECTS AND WINDOWS NT	62
Windows NT and OOP	62
NT Objects and Win32 Objects	63
4.2 I/O REQUEST PACKETS (IRPs)	63
Layout of an IRP	64
Manipulating IRPs	65
4.3 DRIVER OBJECTS	67
Layout of a Driver Object	68
4.4 DEVICE OBJECTS AND DEVICE EXTENSIONS	69
Layout of a Device Object	69
Manipulating Device Objects	70
Device Extensions	71
4.5 CONTROLLER OBJECTS AND CONTROLLER EXTENSIONS	71
Layout of a Controller Object	72
Manipulating Controller Objects	72
Controller Extensions	72
4.6 ADAPTER OBJECTS	74
Layout of an Adapter Object	74
Manipulating Adapter Objects	75
4.7 INTERRUPT OBJECTS	76
Layout of an Interrupt Object	76
Manipulating Interrupt Objects	77
4.8 SUMMARY	77
CHAPTER 5 GENERAL DEVELOPMENT ISSUES	77
5.1 DRIVER DESIGN STRATEGIES	77
Use Formal Design Models	78
Use Incremental Development	79
Use the Sample Drivers	80
5.2 CODING CONVENTIONS AND TECHNIQUES	80
General Recommendations	80
Naming Conventions	81
Header Files	81
Status Return Values	82
NT Driver Support Routines	83
Discarding Initialization Routines	84
Controlling Driver Paging	85
5.3 DRIVER MEMORY ALLOCATION	86
Memory Available to Drivers	86
Working with the Kernel Stack	87
Working with the Pool Areas	87
System Support for Memory Suballocation	88
5.4 UNICODE STRINGS	91
Unicode String Datatypes	91
Working with Unicode	92

5.5 INTERRUPT SYNCHRONIZATION.....	93
The Problem	93
Interrupt Blocking	94
Rules for Blocking Interrupts.....	94
Synchronization Using DDeferred Procedure Calls	95
5.6 SYNCHRONIZING MULTIPLE CPUs.....	95
How Spin Locks Work.....	95
Using Spin Locks	96
Rules for Using Spin Locks	97
5.7 LINKED LISTS	98
Singly-Linked Lists.....	98
Doubly-Linked Lists	99
Removing Blocks from a List	99
5.8 SUMMARY.....	100
CHAPTER 6 INITIALIZATION AND CLEANUP ROUTINES.....	101
6.1 WRITING A DRIVERENTRY ROUTINE	101
Execution Context.....	101
What a DriverEntry Routine Does	102
Initializaig DriverEntry Points	103
Creating Device Objects	103
Choosing a Buffering Strategy.....	104
NT and Win32 Device Names	105
6.2 CODE EXAMPLE: DRIVER INITIALIZATION	105
INIT.C	106
6.3 WRITING REINITIALIZE ROUTINES	113
Execution Context.....	113
What a Reinitializae Routine Does	113
6.4 WRITING AN UNLOAD ROUTINE.....	114
Execution Context.....	114
What an Unload Routine Does.....	114
6.5 CODE EXAMPLE: DRIVER CLEANUP.....	115
UNLOAD.C	115
6.6 WRITING SHUTDOWN ROUTINES.....	118
Execution Context.....	118
What a Shutdown Routine Does	119
Enabling Shutdownb Notification.....	119
6.7 TESTING THE DRIVER.....	119
Testing Procedure.....	120
The WINOBJ Utility	120
6.8 SUMMARY.....	121
CHAPTER 7 HARDWARE INITIALIZATION.....	122
7.1 FINDING AUTO-DETECTED HARDWARE	122
How Auto-Detectoin Works	122
Auto-Detected Hardware and the Registry	123

Querying the Hardware Database	125
What a ConfigCallback Routine Does	127
Using Configuration Data	128
Translating Configuration Data.....	130
7.2 CODE EXAMPLE: LOCATING AUTO-DETECTED HARDWARE.....	130
AUTOCON.C.....	132
7.3 FINDING UNRECOGNIZED HARDWARE	139
Adding Driver Parameters to the Registry	140
Retrieving Parameters from the Registry	140
Other Sources of Device Information	141
7.4 CODE EXAMPLE: QUERYING THE REGISTRY	142
REGCON.C.....	143
7.5 ALLOCATING AND RELEASING HARDWARE	152
How Resource Allocation Works.....	152
How to Claim Hardware Resources.....	153
How to Release Hardware.....	155
Mapping Device Memory	156
Loading Device Microcode.....	157
7.6 CODE EXAMPLE: ALLOCATING HARDWARE	158
RESALLOC.C	158
7.7 SUMMARY.....	162
CHAPTER 8 DRIVER DISPATCH ROUTINES.....	163
8.1 ENABLING DRIVER DISPATCH ROUTINES.....	163
I/O Request Dispatching Mechanism.....	163
Enabling Specific Function Codes	164
Deciding Which Function Codes to Support	165
8.2 EXTENDING THE DISPATCH INTERFACE	165
Defining Private IOCTL Values	167
IOCTL Argument-Passing Methods	167
Writing IOCTL Header Files	169
8.3 WRITING DRIVER DISPATCH ROUTINES	169
Execution Context.....	170
What Dispatch Routines Do	170
Exiting the Dispatch Routine	171
8.4 PROCESSING SPECIFIC KINDS OF REQUESTS	173
Processing Read and Write Requests	173
Processing IOCTL Requests	174
Managing IOCTL Buffers.....	177
8.5 TESTING DRIVER DISPATCH ROUTINES.....	178
Testing Procedure.....	178
Sample Test Program	178
8.6 SUMMARY.....	179

CHAPTER 9 PROGRAMMED I/O DATA TRANSFERS.....	180
9.1 HOW PROGRAMMED I/O WORKS	180
What Happens during Programmed I/O.....	180
Synchronizing Various Driver Routines	181
9.2 DRIVER INITIALIZATION AND CLEANUP	182
Initializing the Start I/O Entry Point	182
Initializing a DpcForlsr Routine.....	183
Connecting to an Interrupt Source	183
Disconnecting from an Interrupt Source	185
9.3 WRITING A START I/O ROUTINE	185
Execution Context.....	185
What the Start I/O Routine Does	186
9.4 WRITING AN INTERRUPT SERVICE ROUTINE (ISR)	186
Execution Context.....	186
What the Interrupt Service Routine Does	187
9.5 WRITING A DPCFORISR ROUTINE	188
Execution Context.....	188
What the DpcForlsr Routine Does	188
Priority Increments.....	189
9.6 SOME HARDWARE: THE PARALLEL PORT	189
How the Parallel Port Works.....	189
Device Registers.....	191
Interrupt Behavior	192
A Driver for the Parallel Port	192
9.7 CODE EXAMPLE: PARALLEL PORT DRIVER	192
XXDRIVER.H	192
INIT.C	193
TRANSFER.C.....	195
9.8 TESTING THE DATA TRANSFER ROUTINES.....	201
Testing Procedure.....	201
9.9 SUMMARY.....	202
CHAPTER 10 TIMERS.....	203
10.1 HANDLING DEVICE TIMEOUTS	203
How I/O Timer Routines Work.....	203
How to Catch Device Timeout Conditions	204
10.2 CODE EXAMPLE: CATCHING DEVICE TIMEOUTS.....	205
XXDRIVER.H	206
INIT.C	206
TRANSFER.C.....	207
TIMER.C.....	209
10.3 MANAGING DEVICES WITHOUT INTERRUPTS	211
Working with Noninterrupting Devices.....	211
How CustomTimerDpc Routines Work.....	212
How to Set Up a CustomTimerDpc Routine.....	213
How to Specify Expiration Times.....	214
Other Uses for CustomTimerDpc Routines	215

10.4 CODE EXAMPLE: A TIMER-BASED DRIVER.....	215
XXDRIVER.H.....	216
INIT.C	216
TRANSFER.C.....	217
10.5 SUMMARY.....	221
CHAPTER 11 FULL-DUPLEX DRIVERS	222
11.1 DOING TWO THINGS AT ONCE	222
Do You need to Process Concurrent IRPs?.....	223
How the Modified Driver Architecture Works	223
Data Structures for a Full-Duplex Driver.....	224
Implementing the Alternate Path	225
11.2 USING DEVICE QUEUE OBJECTS	225
How Device Queue Objects Work.....	225
How to Use Device Queue Objects.....	226
11.3 WRITING CUSTOMDPC ROUTINES	228
How to Use a CustomDpc Routine	228
Execution Cointext.....	229
11.4 CANCELING I/O REQUESTS.....	229
How IRP Cancellation Works.....	230
Synchronization Issues.....	231
What a Cancel Routine Does	232
What a Duispatch Cleanup Routine Does	234
11.5 SOME MORE HARDWARE: THE 16550 UART.....	236
What the 16550 UART Does	236
Device Registers.....	236
Interrupt Behavior	238
11.6 CODE EXAMPLE: FULL-DUPLEX UART DRIVER	239
What to Expect	240
DEVICE_EXTENSION in XXDRIVER.H.....	240
DISPATCH.C.....	241
DEVQUEUE.C	244
INPUT.C	247
ISR.C	249
CANCEL.C	253
11.7 SUMMARY.....	257
CHAPTER 12 DMA DRIVERS	258
12.1 HOW DMA WORKS UNDER WINDOWS NT	258
Hiding DMA Hardware Variaitons with Adapter Objects.....	258
Solving the Scatter/Gather Problem with Mapping Registers	259
Managing I/O Buffers with Memory Descriptor Lists.....	261
Maintaining Cache Coherency	263
Categorizing DMA Drivers.....	265
Limitations of the NT DMA Architecture.....	265
12.2 WORKING WITH ADAPTER OBJECTS.....	266
Fiding the Right Adapter Object	266

Acquiring and Releasing the Adapter Object.....	268
Setting Up the DMA Hardware.....	270
Flushing the Adapter Object Cache	271
12.3 WRITING A PACKET-BASED SLAVE DMA DRIVER.....	272
How Packet-Based Slave DMA Works	272
Splitting DMA Transfers.....	274
12.4 CODE EXAMPLE: A PACKET-BASED SLAVE DMA DRIVER.....	276
XXDRIVER.H	276
REGCON.C.....	277
TRANSFER.C.....	278
12.5 WRITING A PACKET-BASED BUS MASTER DMA DRIVER	285
Setting Up Bus Master Hardware	286
Hardware with Scatter/Gather Support	288
Building Scatter/Gather Lists with IoMapTransfer.....	289
12.6 WRITING A COMMON BUFFER SLAVE DMA DRIVER	291
Allocating a Common Buffer.....	291
Using Common Buffer Slave DMA to Maintain Throughput	292
12.7 WRITING A COMMON BUFFER BUS MASTER DMA DRIVER.....	296
How Common-Buffer Bus Master DMA Works	296
12.8 SUMMARY.....	297
CHAPTER 13 LOGGING DEVICE ERRORS	299
13.1 EVENT-LOGGING IN WINDOWS NT	299
Deciding What to Log.....	299
How Event Logging Works	300
13.2 WORKING WITH MESSAGES.....	301
How Message Codes Work	302
Writing Message Definition Files	303
A Small Example: XXMSG.MC.....	305
Compiling a Message Definition Files.....	307
Adding Message Resources to az Driver	308
Registering a Driver as an Event Source.....	309
13.3 GENERATING LOG ENTRIES.....	310
Preparing a Driver for Error Logging	310
Allocating an Error-Log Packet	311
Logging the Error	312
13.4 CODE EXAMPLE: AN ERROR-LOGGING ROUTINE	313
EVENTLOG.C.....	313
13.5 SUMMARY.....	319
CHAPTER 14 SYSTEM THREADS	320
14.1 SYSTEM THREADS	320
When to Use Threads	320
Creating and Terminating System Threads.....	321
Managing Thread Priority	322
System Worker Threads.....	322

14.2 THREAD SYNCHRONIZATION	323
Time Synchronization	323
General Synchronization	323
14.3 USING DISPATCHER OBJECTS	325
Event Objects	325
Sharing Events between Drivers	327
Mutex Objects	327
Semaphore Objects	329
Timer Objects	330
Thread Objects	331
Variations on the Mutex	332
Synchronization Deadlocks	333
14.4 CODE EXAMPLE: A THREAD-BASED DRIVER	334
How the Driver Works	334
The DEVICE_EXTENSION Structure in XXDRIVER.H	335
The XxCreatDevice Function in INIT.C	336
The XxDispatchReadWrite Function in DISPATCH.C	338
THREAD.C	339
TRANSFER.C	341
14.5 SUMMARY	349
CHAPTER 15 HIGHER-LEVEL DRIVERS.....	350
15.1 AN OVERVIEW OF INTERMEDIATE DRIVERS	350
What Are Intermediate Drivers?	350
Should You Use a Layered Architecture?	351
15.2 WRITING LAYERED DRIVERS	352
How Layered Drivers Work	352
Initialization and Cleanup in Layered Drivers	353
Code Fragment: Connecting to Another Driver	354
Other Initialization Concerns for Layered Drivers	356
I/O Request Processing in Layered Drivers	357
Code Fragment: Calling a Lower-Level Driver	359
15.3 WRITING I/O COMPLETION ROUTINES	360
Requesting an I/O Completion Callback	360
Execution Context	361
What I/O Completion Routines Do	362
Code Fragment: An I/O Completion Routine	363
15.4 ALLOCATING ADDITIONAL IRPs	364
The IRP's I/O Stack Revisited	364
Controlling the Size of the IRP Stack	365
Creating IRPs with IoBuildSynchronousFsdRequest	367
Creating IRPs with IoBuildAsynchronousFsdRequest	368
Creating IRPs with IoBuildDeviceIoControlRequest	369
Creating IRPs from Scratch	371
Setting Up Buffers for Lower Drivers	374
Keeping Track of Driver-Allocated IRPs	375
15.5 WRITING FILTER DRIVERS	376
How Filter Drivers Work	377
Initialization and Cleanup in Filter Drivers	378

What Happens behind the Scenes	380
Making the Attachment Transparent.....	380
15.6 CODE EXAMPLE: A FILTER DRIVER	381
YYDRIVER.H—Driver Data Structures	381
INIT.C—Initialization Code	381
DISPATCH.C—Filter Dispatch Routines	386
COMPLETE.C—I/O Completions Routines	390
15.7 WRITING TIGHTLY COUPLED DRIVERS	394
How Tightly Coupled Drivers Work.....	394
Initialization and Cleanup in Tightly Coupled Drivers.....	395
I/O Request Processing in Tightly Coupled Drivers.....	396
15.8 SUMMARY.....	397
CHAPTER 16 BUILDING AND INSTALLING DRIVERS.....	398
16.1 BUILDING DRIVERS	398
What BUILD Does.....	398
How to Build a Driver.....	400
Writing a SOURCES File	401
Log Files Generated by BUILD	403
Recursive BUILD Operations	403
16.2 MISCELLANEOUS BUILD-TIME ACTIVITIES	404
Using Precompiled Headers	404
Including Version Information in a Driver	405
Including Nonstandard Components in a BUILD.....	407
Moving Driver Symbol Data into .DBG Files	408
16.3 INSTALLING DRIVERS	409
How to Install a Driver by Hand	409
Driver Registry Entries	410
End-User Installation of Standard Drivers	410
End-User Installation of Nonstandard Drivers.....	412
16.4 CONTROLLING DRIVER LOAD SEQUENCE	413
Changing the Driver's Start Value.....	413
Creating Explicit Dependencies between Drivers.....	414
Establishing Global Group Dependencies	415
Controlling Load Sequence within a Group.....	416
16.5 SUMMARY.....	418
CHAPTER 17 TESTING AND DEBUGGING DRIVERS.....	419
17.1 SOME GUIDELINES FOR DRIVER TESTING	419
The General Approach to Testing Drivers	419
Using the Microsoft Hardware Compatibility Tests (HCTs).....	421
17.2 SOME THOUGHTS ABOUT DRIVER BUGS	422
Categories of Driver Errors	422
Reproducing Driver Errors.....	424
Coding Strategies That Reduce Debugging	425
Keeping Track of Driver Bugs.....	425
17.3 READING CRASH SCREENS	426

What Happens When the System Crashes	426
Layout of a STOP Message.....	427
Deciphering STOP Messages.....	429
17.4 AN OVERVIEW OF WINDBG	430
The Key to Source-Code Debugging	430
A Few WINDBAG Commands	431
17.5 ANALYZING A CRASH DUMP	433
Goals of the Analysis	433
Starting the Analysis	433
Tracing the Stack.....	434
Indirect Methods of Investigation	436
Analyzing Crashes with DUMPEXAM	439
17.6 INTERACTIVE DEBUGGING.....	440
Starting and Stopping a Debug Session	440
Setting Breakpoints	441
Setting Hard Breakpoints	442
17.7 WRITING WINDBG EXTENSIONS	442
How WINDBG Extensions Work	442
Initialization and Version-Checking Functions	443
Writing Extension Commands	444
WINDBG Helper Functions.....	445
Building and Using and Extension DLL	446
17.8 CODE EXAMPLE: A WINDBG EXTENSION.....	446
XXDBG.C	446
XXDBG.DEF	451
SOURCES file	451
Sample Output.....	452
17.9 MISCELLANEOUS DEBUGGING TECHNIQUES	453
Leaving Debug Code in the Driver	452
Catching Incorrect Assumptions	453
Using BugCheck Callbacks.....	453
Catching Memory Leaks.....	454
Using Counters, Bits, and Buffers.....	455
17.10 SUMMARY.....	458
CHAPTER 19 DRIVER PERFORMANCE.....	459
18.1 GENERAL GUIDELINES.....	459
Know Where You're Going	459
Get to Know the Hardware.....	460
Explore Creative Driver Designs	460
Optimize Code Creatively.....	461
Measure Everything You Do	461
18.2 PERFORMANCE MONITORING IN WINDOWS NT	462
Some Terminology	462
How Performance Monitoring Works.....	462
How Drivers Export Performance Data	464
18.3 ADDING COUNTER NAMES TO THE REGISTRY	464
Counter Definitions in the Registry	464

Writing LODCTR Command Files	466
Using LODCTR and UNLODCTR	467
18.4 THE FORMAT OF PERFORMANCE DATA	468
Overall Structure of Performance Data.....	468
Types of Counters	470
Objects with Multiple Instances	472
18.5 WRITING THE DATA-COLLECTION DLL.....	474
Contents of the Data-Collection DLL	474
Error Handling in a Data-Collection DLL	476
Installing the DLL	477
18.6 CODE EXAMPLE: A DATA-COLLECTION DLL.....	478
XXPERF.C.....	478
Building and Installing this Example	486
18.7 SUMMARY.....	487
APPENDIX A THE DEVELOPMENT ENVIRONMENT	488
A.1 HARDWARE AND SOFTWARE REQUIREMENTS	488
Connecting the Host and Target.....	489
A.2 DEBUG SYMBOL FILES	490
A.3 ENABLING CRASH DUMPS ON THE TARGET SYSTEM	490
If You Don't Get Any Crash Dump Files	491
A.4 ENABLING THE TARGET SYSTEM'S DEBUG CLIENT	492
APPENDIX B COMMON BUGCHECK CODES	494
B.1 GENERAL PROBLEMS WITH DRIVERS	494
B.2 SYNCHRONIZATION PROBLEMS.....	496
B.3 CORRUPTED DRIVER DATA STRUCTURES.....	496
B.4 MEMORY PROBLEMS.....	498
B.5 HARDWARE FAILURES	500
B.6 CONFIGURATION MANAGER AND REGISTRY PROBLEMS.....	501
B.7 FILE SYSTEM PROBLEMS	503
B.8 SYSTEM INITIALIZATION FAILURES	504
B.9 INTERNAL SYSTEM FAILURES	506
BIBLIOGRAPHY	507
ABOUT THE AUTHOR	509
INDEX.....	511