

# BRIEF OVERVIEW OF THIS BOOK

## CHAPTER DESCRIPTIONS

### **Chapter 1: Adding I/O Devices to a Modern PC**

In this chapter I present an overview of the USB Architecture to ensure that we are all using the same terms in the same way. I discuss the upwards and backwards compatible way that USB 2.0 was designed and I stress the continued development and use of USB 1.1 products. I explain that both I/O device design and PC host software are layered to enable easy support of additional I/O devices. The first range of the I/O devices I cover in this book are implemented as “Human Interface Devices,” or HID, since the operating system already contains all of the software drivers to support this class of device. I cover all other data transfer types, with working examples, in later chapters. I also review the impact of USB on PC host design, and non-PC host design, as giving further impetus to drive the creation of even more USB-based I/O devices.

### **Chapter 2: Close to the Wire**

In this chapter I study the data transfer requests on the USB bus from the bottom up. I look at the signaling scheme on the wires and show that it reveals a packetized structure. I identify the three speeds of the bus (low, full and high) along with the mechanisms that a device uses to identify itself. I explain that sequences of packets are used to generate transactions and that three types of transactions are used during run time (interrupt, bulk, and isochronous), and control transactions are used during enumeration and run time. I mention private host-to-hub transactions for completeness but I do not cover these in detail in this I/O device design book. I then list the available control requests that a PC host can use to interact with an I/O device and a hub device. Finally, I look at a set of design tools that allows direct viewing, detection, and capture of bus packets.

### **Chapter 3: Getting to Know You: Enumeration**

In this chapter I study the reaction of a PC host to a new, unknown I/O device being added to a running system. I detail the enumeration process and note the responses expected of an I/O device. The process flow is logical, although somewhat verbose

from a hardware designer's experience—all information is resident on the I/O device in data tables, called descriptors. This approach makes it easier for the PC host software to support “generic” devices and this, in turn, results in less PC host software for us to develop. I define a block diagram of a “minimum” I/O device.

#### **Chapter 4: Choosing a USB I/O Device**

The protocol-based theory of Chapter 3 is transformed into practical implementations in Chapter 4. I describe the different methods of designing an I/O device and offer guidance on the approaches best suited for a range of applications is presented. A key decision will be the speed of the I/O device (low, full or high) and the tradeoffs and examples in this area are explained. Finally I review some of the turn-key products developed for specific interfacing tasks (such as Ethernet, ATAPI, etc).

#### **Chapter 5: I/O Device Development Environment**

In this chapter I detail the development environment options for a USB I/O device. All of the I/O solutions that I describe will require a combination of skills involving I/O device hardware, I/O device firmware, and PC host software. These are multiple and diverse skills that must be mastered. A team of three typically implements a USB I/O solution, but, with the help of the cookbook methods I describe in this book, a single developer will be able to make a lot of progress. I focus on the processes used to develop I/O device firmware in this chapter. The variety of solutions presented lets you choose the path most suitable for your project.

#### **Chapter 6: PC Host Software**

In this chapter I explain the USB software structure on the PC host via several example programs. The first program, a USB Device Display program, extracts and displays all of the descriptors introduced in the Chapter 3, of all of the devices currently attached via the USB bus. The program does this by searching for devices on the USB bus, so you will learn valuable techniques for the “low-level” USB activity. The second program, a HID Display program, searches for and displays information on the currently installed HID-class devices. In this program, I use a “high-level” file style of access to the I/O devices so that we can focus on the information transferred between the PC host and the I/O device, and not on the details within the USB packets.

## Chapter 7: Design Example: Buttons and Lights

Once we get to this chapter, we have enough information to construct our first USB I/O device. I chose the simple example of writing to a remote 8-bit port and reading from a remote 8-bit port via USB so we can focus on the design **method**. This example is the hardware equivalent of a “Hello World” program. I use the HID driver included with Windows so there is no OS code to write. Attaching to a HID device is a little convoluted but once we have a connection to our I/O device then reading and writing is very easy. I work through the complete design, including hardware implementation, firmware design and implementation and PC host applications software, using the processes developed in Chapters 5 and 6.

## Chapter 8: Completing the Basic Design

Several key design details were omitted in Chapter 7 so that more rapid progress could be made. The example design is **operational**, but it is not **compliant** with the USB Specification. I cover power management in detail in this chapter including an example of remote wakeup. A vendor ID must be obtained from the USB Implementors Forum in order to sell any I/O device as a product. I describe this registration process and the testing methods and tools that are used to ensure compatibility with the USB Specification and interoperability with other devices.

## Chapter 9: Expanding the Basic Design

The working example of Chapter 8 is extended in all possible directions—up and down! I detail the “behind-the-scenes” operation of a HID device such that an appreciation of how best to improve our I/O device can be understood. I implement multiple interfaces, multiple reports, and larger amounts of data using a variety of working examples. Some applications require the I/O device to be electrically isolated from the PC Host—I present two alternate approaches. The maximum length of a USB cable segment is 5 metres so, when using the maximum depth of hubs, the I/O device is constrained to be within 30 metres of the PC Host—I discuss the requirements for a “bus-extender” and present some alternate solutions. Finally I implement a cost-reduction exercise to tune the prototype into a production product, including a change to a different USB device.

## Chapter 10: Building I/O Bridges

I study the options of bridging existing standard interfaces to USB in this chapter. I employ the techniques developed in the previous chapters on RS232, ISA, I2C, and IrDA interfaces. The bridge to an RS-232 modem, for example, contains many redundant components, so this design example is optimized to produce a cheaper and higher-performance, USB-based modem. I look at a variety of custom interfaces which all inherit the improved capabilities and ease-of-use of USB.

## Chapter 11: Moving a Lot of Data

Most of the examples so far have been moving small amounts of intermittent data using a HID interface. Many applications require a large amount of data to be moved and this chapter focuses on bulk transactions. Windows does not include a simple BlockIO Class Driver so, in this chapter, I write a WDM device driver and demonstrate it with several bulk transfer examples. I also develop matching I/O device firmware for this custom interface. I move on to present the structure of a WDM Device Driver and use several filter examples to best demonstrate the techniques and approach that is needed. These examples include a filter that Displays Usb Enumeration Transactions (DUET), a filter that can modify descriptors at run time (TwoKeyboards) and a filter driver that can initialize a custom I/O device (LoadEz). There is a comprehensive bulk transfer driver called the Mass Storage Class driver so I develop I/O device firmware to interact with this class driver.

## Chapter 12: I Like the Sound of That!

In this chapter we discover that handling real-time data, such as sound, is not much more difficult than handling other large amounts of data. Following an investigation of “digital audio,” I have detailed the mechanism for supporting sound in USB via the full implementation of several design examples: high-quality audio output using speakers, and lower-quality, bidirectional audio using a telephone. You discover that sound is an easy attribute to add to many I/O devices.

## Chapter 13: I Can See You!

As far as USB is concerned, digital video is just like digital audio except there is a lot more data! In this chapter I study the real amount of throughput that USB 1.1 can sustain and conclude that video data must be compressed to be usable. I develop an example that accepts video from a USB camera and displays it on the PC screen. This software is then augmented to add frame-to-frame comparisons and thus creates a “digital, motion-detector VCR.” The higher bandwidth of USB 2.0 enables higher quality video and I present full details of a DirectShow example.

## Chapter 14: Hubs—The Inside Story

Adding a hub to a PC host is easy since the Windows operating system supports a range of plug-and-play PCI add-in cards. In this chapter I look at the added capabilities that can be implemented if a hub is included inside an I/O device. I work through a monitor example with gradually increasing capabilities.

## **Chapter 15: Portable and Handheld Designs**

After its initial success in the PC arena, USB is now being added to consumer devices such as portable and handheld appliances. I present a variety of client solutions including digital still cameras, personal data assistants (PDA), and an MP3 player; I describe an overview of each design then detail specific USB implementation details.

## **Chapter 16: But I Don't Do Windows!**

All of the design examples that I have presented so far have been based on a Windows WDM platform. USB support was first integrated into Windows 98 and is now supported in Windows 2000. Many other operating systems, including Linux, BeOS, Mac OS 9 and X and WinCE also support USB. Many real-time operating systems include USB support and a range of implementations is investigated. I rework two of the examples under these operating systems to demonstrate the similarity in approaches.

## **Appendix A About the CD-ROM**

This section references material that is on the CD-ROM. The CD-ROM includes a hypertext document with live links to vendors' sites for all of the products mentioned in this book.

## **Appendix B USB Microcontroller Interfacing Ideas**

The focus of this book is the "USB side" of the microcontroller. I learnt many useful techniques while implementing the examples and this appendix includes a variety of sensor and transducer interfacing schemes, with the matching firmware, that I used.

---

# Contents

## Chapter 1 Adding I/O Devices to a Modern PC

The Reasons for This Book .....	2
Hands-on Examples .....	2
How Much Technical Background Do You Need? .....	3
Focus of the Text .....	3
The Modern PC: A Short History .....	4
The PCI Bus .....	4
Easy to Use and Low Cost .....	5
USB Terminology .....	7
PC Host .....	10
USB Cable .....	11
Hub Device .....	13
I/O Device .....	14
Adding USB to an “Old” PC .....	16
Impact of USB on PC Host .....	17
Impact of USB on Embedded PCs and Other Devices .....	21
Chapter Summary .....	23

## Chapter 2 Close to the Wire

Differential Signaling .....	25
The Fundamental Packet .....	26
Different Packet Types .....	28
Start-of-Frame (SOF) Token Packet .....	28
IN, OUT, and SETUP Token Packets .....	30
Data Transfer Packets .....	31
Handshake Packets .....	32
Special Packets .....	33
Building a Transaction .....	34
Interrupt Transactions .....	35
Bulk Transactions .....	37
Isochronous Transactions .....	37
Control Transactions .....	39
PC Host Requests .....	44
Error Handling .....	47
Chapter Summary .....	47

### Chapter 3 Getting to Know You: Enumeration

Device Detection .....	50
Enumeration Steps .....	52
High-Speed Handshake .....	53
Device Descriptor .....	54
Configuration Descriptor .....	56
Interface Descriptor .....	57
Defining a Human Interface Device .....	58
HID Descriptor .....	59
Report Descriptor .....	60
Choosing a Device Driver .....	62
Minimum I/O Device .....	63
Enhancing the Minimum Device .....	64
Alternate-Speed Descriptors .....	66
Descriptors for a Feature-Rich I/O Device .....	67
Additional I/O Device Responsibilities .....	69
Chapter Summary .....	71

### Chapter 4 Choosing a USB I/O Device

Putting the Pieces Together .....	74
Start with a Transceiver . . . . .	74
. . . Add a Serial Interface Engine . . . . .	75
. . . Add a Protocol Controller . . . . .	80
. . . Finally, Add Real-World I/O . . . . .	82
Custom ASIC .....	87
What is USB Semiconductor Intellectual Property (SIP)? .....	88
Various kinds of USB SIP are available .....	88
What you should look for in USB SIP .....	88
What you should look for in a USB SIP vendor .....	90
Fixed-Function Devices .....	91
Selection Criteria .....	92
Chapter Summary .....	93

### Chapter 5 I/O Device Development Environment

Development Tools .....	96
Target Implementations .....	98
A Bond-Out Example .....	99
A Debug Monitor Example .....	102
An Integrated Debug Monitor Example .....	103
A USB Peripheral Example .....	104
A USB ASIC Example .....	105

Firmware Development Tools .....	106
PC Host Development Tools .....	108
USB Software Tools .....	109
USB View .....	109
Hidview .....	110
HID Descriptor Tool .....	111
Single-Stepping of USB Enumeration .....	112
USB Hardware Tools .....	114
Chapter Summary .....	120

## Chapter 6 PC Host Software

Visual Basic Review .....	123
Example 1: USB Device Display .....	126
Step 1: Human Interface Design .....	128
Step 2: Initializing Program .....	130
Step 3: Enumerating a Host Controller .....	131
Step 4: Descriptor Display .....	135
Example Summary .....	138
Example 2: Display Human Interface Devices .....	138
Step 1: Human Interface Design .....	138
Step 2: Initializing Program .....	139
Step 3: Displaying HID Information .....	141
Example Summary .....	141
Visual C++ Review .....	142
Example 1: USB Device Display .....	143
Step 1: Human Interface Design .....	144
Step 2: Initializing Program .....	145
Step 3: Enumerating a Host Controller .....	146
Step 4: Descriptor Data Collection .....	150
Example Summary .....	152
Example 2: Display Human Interface Devices .....	152
Step 1: Human Interface Design .....	153
Step 2: Initializing Program .....	153
Step 3: Displaying HID Information .....	155
Example Summary .....	155

## Chapter 7 Design Example: Buttons and Lights

Design Example: Buttons and Lights .....	158
Step 1: Designing the Hardware .....	159
Step 2: Completing the Descriptors .....	161
Step 3: Implementing Microcontroller Code .....	163

Step 4: Attaching an Empty USBSIMM to the PC Host .....	180
Step 5: Debugging I/O Device Hardware and Software .....	181
Step 6: Writing the PC Host Application .....	184
Design Summary .....	185
Chapter Summary .....	186

## Chapter 8 Completing the Basic Design

Handling Power Requirements .....	187
Responding to a Suspend .....	188
Generating a Wakeup Event .....	189
Getting a Vendor ID .....	190
USB IF Compliance Overview .....	191
USB IF Testing Details .....	193
Interoperability .....	193
Interoperability Additions for High-Speed Testing .....	195
USBCheck .....	196
USBCheck Additions for High-Speed Testing .....	198
Full- and Low-Speed Signal Quality .....	198
Inrush Current .....	201
Power Consumption .....	201
High-Speed Electrical Testing .....	201
High-Speed Electrical Testing Equipment .....	202
Test Fixture .....	203
High-Speed Signal Quality .....	204
Receiver Sensitivity .....	206
Microsoft WHQL USB Testing Overview .....	210
Microsoft WHQL Testing Details .....	211
Common Device Problems Uncovered During Testing .....	213
Chapter Summary .....	214

## Chapter 9 Expanding the Basic Design

What We Have Built .....	215
Extending the Reports .....	220
Adding Reports .....	221
Adding Interfaces .....	222
Changing the Report Format .....	226
Cutting the USB Cable .....	228
Using a High-Level Language .....	232
Changing the USB Microcontroller .....	233
Changing the Real-World I/O Interface .....	238
USB-to-Smart Card Design Example .....	239
USB-to-RS232 Connection .....	240
Chapter Summary .....	241

## Chapter 10 Building I/O Bridges

Design of a USB-to-RS232 Bridge	243
Design of a Serial Communications Peripheral	249
A Look at Communications Standards	250
Direct Line Control Modem Example	251
Removing the RS232 Connection	252
Moving Command Set Interpretation to the PC Host	253
Removing the DSP	254
Can Anything Else Be Removed or Simplified?	254
Results of Design Optimization	257
Migration from ISA	260
IN and OUT Are Special	263
Building an ISA Card	263
Plug and Play ISA	264
USB-to-ISA Bridge	265
Parallel Device Examples	269
Floppy Disk Drive	269
Hard Disk Drive	272
SCSI Devices	273
USB-to-LAN Bridges	274
Bridging to Other PCs	275
Chapter Summary	277

## Chapter 11 Moving a Lot of Data

A Block I/O Example	281
Step 1: Design the Hardware	281
Step 2: Complete the Descriptors	281
Step 3: Implement Microcontroller Code	283
Step 4: The BlockIO Device Driver	287
Step 5: The Application Program	297
Step 6: Installing our BlockIO Device	300
Welcome to the Kernel World	302
EZ-USB Autoloader	302
A Filter Driver	306
Display USB Enumeration Transactions	309
Two Keyboards	311
Using a Windows-Supplied Device Driver	313
Chapter Summary	314

## Chapter 12 I Like the Sound of That!

Describing Hardware Using Descriptors	315
Audio Control Interface	315
Audio Streaming Interface	318

Implementing the I/O Device Firmware .....	321
Upgrading to 16-bit Stereo .....	324
Dedicated Speaker Solution .....	326
Audio Input via a Microphone .....	328
CD Quality Sound Input and Output .....	329
TUSB3200 Architecture .....	330
TUSB3200 Evaluation Module .....	331
Development Tools .....	332
Getting Started with TUSB3200 Programming .....	333
Sample Pseudocode for STC Initialization .....	333
An Internet Telephone .....	336
Desktop Business Audio .....	338
MIDI Overview .....	341
MIDI Protocol .....	342
MIDI File Format .....	343
MIDI Synthesizers .....	343
MIDI Hardware Interface .....	344
USB-Enhanced MIDI .....	346
Chapter Summary .....	347

## Chapter 13 I Can See You!

Digital Video Application .....	350
Sizing Video Data .....	351
Video Compression Is Essential .....	352
Example 1: Videoconferencing Camera .....	354
Example 2: Composite Video .....	356
Example 3: Digital Video Creation .....	358
Windows Video Software—DirectShow .....	361
DirectShow Projects .....	363
USB-Enabled Video Applications .....	368
Digital Microscope .....	368
Biometrics .....	369
Fingerprint Identification .....	370
IriScan Application .....	371
Chapter Summary .....	372

## Chapter 14 Hubs—The Inside Story

Full-/Low-Speed Hub .....	373
Hub Repeater .....	374
Hub Controller .....	376
Power Control .....	378
Full-/Low-Speed Hub Summary .....	380
High-/Full-/Low-Speed Hub .....	381

Building a Compound Device .....	386
Design Example .....	387
Step 1: Adding a Hub .....	389
Step 2: Adding I/O Devices .....	391
Step 3: Extensible Design .....	392
Chapter Summary .....	394

## Chapter 15 Portable and Handheld Designs

A Security Key .....	396
Remote Data Collection .....	398
More Portable Data Storage .....	399
Data Storage Technologies .....	400
Accessing Data Storage Devices .....	404
MP3 Player .....	406
Digital Still Camera .....	410
Personal Digital Assistant .....	416

## Chapter 16 But I Don't Do Windows!

The Linux Kernel .....	421
USB Features .....	423
Buttons-and-Lights Example .....	423
The VxWorks Kernel .....	426
Real-Time Determinism .....	428
High-Performance Microkernel Design .....	430
Scaleable Run-Time Software .....	430
USB in VxWorks .....	431
Components Provided in Source Code .....	432
Developing a Client Driver for an I/O Device under VxWorks .....	434
EEPROM Loader .....	436
Chapter Summary .....	437

## Appendix A About the CD-ROM .....

## Appendix B USB Microcontroller Interfacing Ideas .....

The I2C Interface .....	441
The I2C Specification .....	442
I2C Summary .....	446
Thermometer Applications .....	447
Example 1: Reading Temperatures .....	447
Example 2: Adding Temperature Limits .....	450
Example 3: Using a Multidrop Thermometer .....	451
Thermometer Applications Summary .....	453

- Infrared Subsystems ..... 453
  - PC Industry IR Standards ..... 454
  - Example 1: IrDA Data ..... 454
  - Example 2: IrDA Control ..... 458
  - Example 3: Consumer Industry IR ..... 462
  - Infrared Subsystems Summary ..... 465
- Connecting to the Real World ..... 466
  - Output Signal Amplification ..... 466
- Controlling a Motor ..... 468
  - Example 1: Stepper Motor ..... 468
  - Example 2: DC Motor ..... 473
  - Example 3: Lighting Panel ..... 474
- Analog Signal Interfacing ..... 478
  - Analog Conversion Examples ..... 478
  - Sensor Inputs ..... 483
- Data Acquisition and Instrumentation ..... 485
  - USB Module Examples ..... 486
  - USB Industrial Series Example ..... 490
- Chapter Summary ..... 492
- Index** ..... 493

---

## PREFACE TO THE SECOND EDITION

There were two equally driving forces that resulted in this second edition of *USB Design By Example*. The first was the advance in USB technology; not only the higher speed of USB2.0, but the many new and exciting developments with USB1.1 products. USB has broken out of its “desktop PC”-centric roots and is now being employed in portable devices, consumer electronics and servers. The 480 Mbps data transfer rate introduced by USB 2.0 will also extend the application range upward to include high performance videoconferencing cameras, printers, and scanners. I have nothing but praise for the engineering group that defined USB 2.0 to be upwards and backwards compatible—all existing USB 1.1 devices will continue to operate unchanged, with the same cables and connectors! New USB 2.0 hubs will support three data rates: low, full, and high speed (1.5, 12, and 480 Mbps) and will allow new USB 2.0 I/O devices that can take advantage of this forty-fold speed improvement. The beauty and elegance of this solution is highlighted by the *unchanged* end-user model—there is *no need* for the end-user to even know that there was once a 1.1 spec and is now a 2.0 spec. “New” USB 2.0 operates just the same as “old” USB 1.1. New lower-cost low-speed devices will continue to be developed, which will further increase the dynamic range of USB-based I/O solutions.

The second driving force was the stream of hundreds of e-mails that I received from readers of the first edition. Many of these began, “But all I want to do is . . .”—so I have added many more examples that can be easily expanded. A recurring theme—“focus on the USB-specific aspects, assume we know how to interface a microcontroller to sensors and transducers”—is addressed by adding more USB material and referring to other microcontroller design books for the “real-world” interfaces. Readers wanted more software and more explanation of that software, so I have augmented the Windows material, and added Linux and WindRiver real-time material. Device drivers are now covered along with additional applications code examples. The focus of the book is still “how to add I/O devices to a modern PC,” but I do look inside hubs to explain their operation so that you can design better I/O devices. Thank you again for the many e-mails—keep them coming! I will continue to post additional examples on the companion website at <http://www.intel.com/intelpress/usb>.

Happy Developing!     *John* (john.hyde@intel.com)