Are your company’s technical training needs being addressed in the most effective manner?

MindShare has over 25 years experience in conducting technical training on cutting-edge technologies. We understand the challenges companies have when searching for quality, effective training which reduces the students’ time away from work and provides cost-effective alternatives. MindShare offers many flexible solutions to meet those needs. Our courses are taught by highly-skilled, enthusiastic, knowledgeable and experienced instructors. We bring life to knowledge through a wide variety of learning methods and delivery options.

training that fits your needs

MindShare recognizes and addresses your company’s technical training issues with:

• Scalable cost training
• Customizable training options
• Just-in-time training
• Overview and advanced topic courses
• Reducing time away from work
• Training delivered effectively globally
• Concurrently delivered multiple-site training

MindShare training courses expand your technical skillset

- PCI Express 2.0®
- Intel Core 2 Processor Architecture
- AMD Opteron Processor Architecture
- Intel 64 and IA-32 Software Architecture
- Intel PC and Chipset Architecture
- PC Virtualization
- USB 2.0
- Wireless USB
- Serial ATA (SATA)
- Serial Attached SCSI (SAS)
- DDR2/DDR3 DRAM Technology
- PC BIOS Firmware
- High-Speed Design
- Windows Internals and Drivers
- Linux Fundamentals
- ... and many more.

All courses can be customized to meet your group’s needs. Detailed course outlines can be found at www.mindshare.com

*PCI Express ® is a registered trademark of the PCISIG
Engage MindShare

Have knowledge that you want to bring to life? MindShare will work with you to “Bring Your Knowledge to Life.” Engage us to transform your knowledge and design courses that can be delivered in classroom or virtual classroom settings, create online eLearning modules, or publish a book that you author.

We are proud to be the preferred training provider at an extensive list of clients that include:

- ADAPTEC
- AMD
- AGILENT TECHNOLOGIES
- APPLE
- BROADCOM
- CADENCE
- CRAY
- CISCO
- DELL
- FREESCALE
- GENERAL DYNAMICS
- HP
- IBM
- KODAK
- LSI LOGIC
- MOTOROLA
- MICROSOFT
- NASA
- NATIONAL SEMICONDUCTOR
- NETAPP
- NOKIA
- NVIDIA
- PLX TECHNOLOGY
- QLOGIC
- SIEMENS
- SUN MICROSYSTEMS
- SYNOPSYS
- TI
- UNISYS
FireWire® System Architecture, Second Edition
## Contents

### About This Book

The MindShare Architecture Series ................................................................. 1  
Cautionary Note ................................................................................................... 2  
Organization of This Book .................................................................................. 2

- **Part One: Introduction to FireWire (IEEE 1394)** ........................................ 2
  - Chapter 1: Why FireWire? ........................................................................... 2  
  - Chapter 2: Overview of the FireWire Architecture ......................................... 2
- **Part Two: Serial Bus Communications** ....................................................... 3
  - Chapter 3: Communication Model .................................................................. 3  
  - Chapter 4: Communications Services ............................................................ 3
  - Chapter 5: Cables & Connectors ..................................................................... 3
  - Chapter 6: The Electrical Interface ................................................................. 3
  - Chapter 7: Arbitration ..................................................................................... 3
  - Chapter 8: Asynchronous Packets .................................................................. 3
  - Chapter 9: Isochronous Packets .................................................................... 3
  - Chapter 10: PHY Packet Format ..................................................................... 4
  - Chapter 11: Link to PHY Interface ................................................................. 4
  - Chapter 12: Transaction Retry ....................................................................... 4
- **Part Three: Serial Bus Configuration** .......................................................... 4
  - Chapter 13: Configuration Process .................................................................. 4  
  - Chapter 14: Bus Reset (Initialization)............................................................... 4
  - Chapter 15: Tree Identification ...................................................................... 4
  - Chapter 16: Self Identification ....................................................................... 4
- **Part Four: Serial Bus Management** ............................................................... 5
  - Chapter 17: Cycle Master .............................................................................. 5  
  - Chapter 18: Isochronous Resource Manager .................................................. 5
  - Chapter 19: Bus Manager .............................................................................. 5
  - Chapter 20: Bus Management Services .......................................................... 5
- **Part Five: Registers and Configuration ROM** ............................................. 5
  - Chapter 21: CSR Architecture ....................................................................... 5  
  - Chapter 22: PHY Registers .......................................................................... 5
  - Chapter 23: Configuration ROM ..................................................................... 5
- **Part Six: Power Management** ..................................................................... 6
  - Chapter 24: Introduction to Power Management ........................................... 6
  - Chapter 25: Cable Power Distribution ............................................................ 6
  - Chapter 26: Suspend & Resume ..................................................................... 6
  - Chapter 27: Power State Management ............................................................ 6
- **Appendix** ..................................................................................................... 6  
  - Example 1394 Chip Solutions ....................................................................... 6

### Target Audience .......................................................................................... 7

### Prerequisite Knowledge ................................................................................... 7
## Contents

Documentation Conventions ................................................................................................... 7  
  Labels for Multi-byte Blocks ............................................................................................ 7  
  Hexadecimal Notation .................................................................................................... 8  
  Binary Notation ............................................................................................................ 8  
  Decimal Notation .......................................................................................................... 8  
  Bit Versus Byte Notation ............................................................................................... 9  

Identification of Bit Fields (logical groups of bits or signals) ........................................... 9  

Visit Our Web Page ........................................................................................................... 10  
We Want Your Feedback ................................................................................................. 10  

---  

### Part One

Introduction to FireWire (IEEE 1394a)

### Chapter 1: Why FireWire?

Overview ....................................................................................................................... 13  

Motivations Behind FireWire Development ....................................................................... 13  
  Inexpensive Alternate to Parallel Buses ....................................................................... 14  
  Plug and Play Support .................................................................................................. 14  
  Eliminate Host Processor/Memory Bottleneck ............................................................ 14  
  High Speed Bus with Scalable Performance ................................................................ 15  
  Support for Isochronous Applications ........................................................................ 15  
  BackPlane and Cable Environments .............................................................................. 15  
  Bus Bridge ..................................................................................................................... 15  

1394 Applications ............................................................................................................. 16  

IEEE 1394 Refinements ................................................................................................... 16  

Primary Features ............................................................................................................ 17  

---  

### Chapter 2: Overview of the IEEE 1394 Architecture

IEEE 1394 Overview ....................................................................................................... 19  

Specifications and Related Documents ........................................................................... 20  
  IEEE 1394-1995 and the IEEE 1394a Supplement ....................................................... 20  
  IEEE 1394.B .................................................................................................................. 21  
  Unit Architecture Specifications .................................................................................... 21  

IEEE 1394 Topology ....................................................................................................... 23  
  Multiport Nodes and Repeaters ............................................................................... 23  
  Configuration ................................................................................................................ 23  
  Peer-To-Peer Transfers ............................................................................................... 24  
  Device Bay .................................................................................................................... 24
## Contents

The ISO/IEC 13213 Specification .......................................................................................... 24
  Node Architecture ........................................................................................................ 25
  Address Space ............................................................................................................. 28
  Transfers and Transactions ........................................................................................ 30
    Asynchronous Transfers ........................................................................................ 30
    Isochronous Transfers ........................................................................................ 32
  Control and Status Registers (CSRs) ............................................................................. 33
  Configuration ROM ...................................................................................................... 33
  Message Broadcast ....................................................................................................... 34
  Interrupt Broadcast ..................................................................................................... 34
  Automatic Configuration ............................................................................................. 34

### Part Two

**Serial Bus Communications**

### Chapter 3: Communications Model

Overview......................................................................................................................... 37

Transfer Types................................................................................................................ 39
  Asynchronous ............................................................................................................. 40
  Isochronous ............................................................................................................... 41

The Protocol Layers ....................................................................................................... 42
  Bus Management Layer ............................................................................................ 44
  Transaction Layer ...................................................................................................... 45
    Transaction Layer Services ..................................................................................... 45
  Link Layer ................................................................................................................... 47
    Split Transactions ................................................................................................... 49
    Concatenated Transactions ...................................................................................... 51
    Unified Transactions ............................................................................................... 52
  Physical Layer ............................................................................................................ 53
    Twisted Pair Signaling ............................................................................................ 54
      Bus Configuration ................................................................................................. 55
      Arbitration ........................................................................................................... 55
      Data Transmission ............................................................................................... 56
    Power Pair ............................................................................................................... 56
    Packet-Based Transactions ..................................................................................... 56
      Asynchronous Packets .......................................................................................... 56
      Isochronous Packet ............................................................................................. 58
    Port Repeater ........................................................................................................... 59
## Contents

**A Sample Asynchronous Transaction** ................................................................. 60
  The Request ........................................................................................................... 61
  The Response .......................................................................................................... 61

**An Example Isochronous Transaction** .............................................................. 63

### Chapter 4: Communications Services

Overview .................................................................................................................. 65

**Anatomy of Asynchronous Transactions** .......................................................... 66
  The Request Subaction ................................................................................................... 66
    Initiating the Transaction (The Request) ................................................................. 69
      Transaction Layer .................................................................................................. 69
      The Link Layer ...................................................................................................... 69
      The PHY Layer ..................................................................................................... 70
    Receiving the Request (The Indication) ................................................................. 71
      Physical Layer ...................................................................................................... 71
      Link Layer ............................................................................................................. 71
      Transaction Layer .................................................................................................. 72
      The Acknowledgment ............................................................................................. 72
  Response Subaction .................................................................................................. 73
    Reporting the Results (The Response) ................................................................... 74
      Transaction Layer Response ................................................................................ 74
      Link Layer Response ............................................................................................ 75
      PHY Layer Response ........................................................................................... 75
    Response Reception ................................................................................................ 76
      Physical Layer ...................................................................................................... 76
      Link Layer ............................................................................................................. 76
      Transaction Layer .................................................................................................. 76
      The Acknowledgment ............................................................................................. 77
  Transaction Label ...................................................................................................... 78

**Anatomy of Isochronous Transactions** ............................................................ 78
  Setting Up Isochronous Transactions ....................................................................... 79
  Maintaining Synchronization .................................................................................... 79
  Isochronous Transactions ....................................................................................... 80
  Isochronous Transaction Initiation & Reception ..................................................... 80
    Initiating the Transaction ....................................................................................... 81
      Link Layer ............................................................................................................. 81
      The PHY Layer ..................................................................................................... 81
    Transaction Reception ............................................................................................. 82
      Physical Layer ...................................................................................................... 83
      Link Layer ............................................................................................................. 83
Contents

Chapter 5: Cables & Connectors

Cable and Connector Types ................................................................. 85

6-pin Connector (1394-1995) ................................................................. 86
Make First/Break Last Power Pins ....................................................... 87
Optional 4-pin Connector (1394a supplement) ................................. 87
Positive Retention ............................................................................. 88

Cable Characteristics ........................................................................ 89

6-Conductor Cables ........................................................................... 89
4-Conductor Cables ........................................................................... 90

Device Bay .......................................................................................... 92

Chapter 6: The Electrical Interface

Overview ............................................................................................ 95

Common Mode Signaling ................................................................. 96
Differential Signaling ........................................................................ 96

Recognition of Device Attachment and Detachment ...................... 97
IEEE 1394-1995 Device Attachment/Detachment ............................ 97
IEEE 1394a Device Attachment/Detachment ..................................... 99
Bus Idle State .................................................................................... 100

The Port Interface ............................................................................... 101

Differential Signal Specifications ..................................................... 104

Arbitration Signaling ......................................................................... 105
Line State Signaling (1, 0, and Z) ...................................................... 105
Line State Detection .......................................................................... 107

Reset Signaling ................................................................................ 110

Line States During Configuration ...................................................... 110
Line States During Normal Arbitration ............................................ 112

Starting and Ending Packet Transmission ....................................... 114
Dribble Bits .................................................................................... 116

Port State Control ............................................................................. 116

Speed Signaling ................................................................................ 117
High Speed Devices Slowed Due to Topology ................................ 118
Devices of Like Speed Directly Connected .................................... 118
Speed Signaling Circuitry ................................................................. 119

Data/Strobe Signaling ...................................................................... 122
NRZ Encoding ................................................................................ 123
Data-Strobe Encoding ..................................................................... 124

Gap Timing ....................................................................................... 125

Cable Interface Timing Constants ................................................... 128
Suspend/Resume ............................................................................. 133
## Contents

Cable Power .................................................................................................................... 133  
  Cable Power Requirements ......................................................................................... 134  
  Power Class .................................................................................................................. 135  
  Power Distribution ....................................................................................................... 137  
  Bus Powered Nodes ..................................................................................................... 138  

### Chapter 7: Arbitration

Overview ....................................................................................................................... 141  
Arbitration Signaling ..................................................................................................... 142  
Arbitration Services ..................................................................................................... 145  
Asynchronous Arbitration ............................................................................................ 147  
  Fairness Interval .......................................................................................................... 147  
  Arbitration Enable Bit ............................................................................................. 147  
  Fair Arbitration Service ........................................................................................... 148  
  Arbitration Reset Gap .............................................................................................. 148  
  The Acknowledge Packet and Immediate Arbitration Service ................................... 150  
Isochronous Arbitration ............................................................................................... 150  
  Cycle Start and Priority Arbitration ........................................................................ 152  
Combined Isochronous and Asynchronous Arbitration .............................................. 152  
  Cycle Start Skew ........................................................................................................ 152  
1394a Arbitration Enhancements .................................................................................. 157  
  Acknowledge Accelerated Arbitration .................................................................. 157  
  Fly-by Arbitration ..................................................................................................... 158  
  Acceleration Control ............................................................................................... 159  
  Priority Arbitration Service ..................................................................................... 161  
Summary of Arbitration Types .................................................................................... 163  

### Chapter 8: Asynchronous Packets

Asynchronous Packets .................................................................................................. 165  
  Data Size ...................................................................................................................... 167  
  Write Packets ............................................................................................................... 168  
Asynchronous Stream Packet ....................................................................................... 174  
Read Packets ................................................................................................................ 175  
Lock Operations .......................................................................................................... 183  
  Lock Request Packet .................................................................................................. 183  
  Lock Transaction Types (Extended t_code Field) .................................................. 186  
  Data Block Length During Lock Request ............................................................... 187  
  Lock Response Packet ............................................................................................. 188  
Response Codes .......................................................................................................... 190  
Acknowledge Packet .................................................................................................... 191  
Asynchronous Transaction Summary ......................................................................... 193
# Contents

Write Transactions ................................................................. 193  
Summary of Read and Lock Transactions .................................. 194  
Cycle Start Packet .................................................................. 195

## Chapter 9: Isochronous Packet
Stream Data Packet ................................................................. 199  
Isochronous Data Packet Size .................................................. 201  
Isochronous Transaction Summary .......................................... 202

## Chapter 10: PHY Packet Format
Overview .................................................................................. 205  
PHY Packet Format .................................................................. 206  
Self-ID Packets .......................................................................... 207  
  Self-ID Packet Zero ................................................................. 207  
  Self-ID Packets One, Two, and Three (1394-1995) .................... 210  
  Self-ID Packets One and Two (1394a) ..................................... 211  
Link-on Packet .......................................................................... 212  
PHY Configuration Packet ....................................................... 213  
  Force Root Node .................................................................. 213  
  Gap Count Optimization ....................................................... 214  
Extended PHY Packets ............................................................. 214  
  Ping Packet ........................................................................ 214  
  Remote Access Packet ......................................................... 215  
  Remote Reply Packet ............................................................ 216  
  Remote Command Packet ..................................................... 217  
  Remote Confirmation Packet .................................................. 218  
  Resume Packet .................................................................... 219

## Chapter 11: Link to PHY Interface
Overview .................................................................................. 221  
The Interface Signals ............................................................... 222  
Sharing the Interface ............................................................... 224  
  PHY Initiated Transfers ......................................................... 224  
    Idle State .......................................................................... 225  
    Status State ...................................................................... 225  
    Receive State ..................................................................... 225  
    Grant State ....................................................................... 225  
  Link Initiated Transfers ......................................................... 226  
Determining Transfer Rate Between Link and PHY .................... 227  
Powering the Link .................................................................... 228  
Packet Transmission ............................................................... 228
Contents

Link Issues Request ........................................................................................................ 228
  Request Types .......................................................................................................... 231
  Speed of Link to PHY Data Transmission ............................................................. 231
  When Can the Link Issue a Request? ..................................................................... 232
  PHY Behavior When a Packet Request is Received ............................................. 234
Receiving Packets ........................................................................................................ 236
  Speed of PHY to Link Data Transmission ............................................................. 237
PHY Reports Status ...................................................................................................... 238
  ARB_RESET_GAP.................................................................................................... 239
  SUBACTION_GAP ................................................................................................. 239
  BUS_RESET_START ............................................................................................... 239
  PHY_INTERRUPT ................................................................................................. 240
Accelerated Arbitration Control ..................................................................................... 240
Accessing the PHY Registers ....................................................................................... 241
  PHY Register Reads ............................................................................................. 242
    When Can a Register Read Request Be Issued? ............................................... 243
    PHY Behavior When a Register Read Request is Received ............................... 243
    Register Contents Returned by PHY ................................................................. 243
  PHY Register Writes .............................................................................................. 244
    When Can a PHY Register Write Request Be Issued? .................................... 244
    PHY Behavior When a Register Write Request is Received ............................... 245
Electrical Isolation Between PHY and Link .................................................................. 245

Chapter 12: Transaction Retry
Overview ....................................................................................................................... 247
Busy Retry .................................................................................................................... 248
  The First Packet Transmission Attempt .............................................................. 248
  Single Phase Retry ................................................................................................. 249
    Sending-Node Retry Behavior (Outbound Retry) .............................................. 249
    Receiving-Node Retry Behavior (Inbound Retry) ............................................... 250
  Dual Phase Retry .................................................................................................... 251
    Sending-Node Retry Behavior (Outbound Retry) .............................................. 252
    Receiving-Node Retry Behavior (Inbound Retry) ............................................... 255
Transactions Errors ..................................................................................................... 259
  Packet Transmission Errors .................................................................................. 259
  Packet Error Handling Summary .......................................................................... 262
Part Three
Serial Bus Configuration

Chapter 13: Configuration Process
Overview................................................................................................................................. 265
Bus Initialization (Bus Reset) .............................................................................................. 267
Tree Identification (The Family Tree) ................................................................................ 268
Self Identification .................................................................................................................. 270
Bus Management................................................................................................................. 272

Chapter 14: Bus Reset (Initialization)
Overview................................................................................................................................. 273
Sources of Bus Reset........................................................................................................... 274
Power Status Change....................................................................................................... 274
Bus Reset Signaled by Attached Node.......................................................................... 274
Node Attachment or Removal ....................................................................................... 275
MAX_ARB_STATE_TIME Expires................................................................................ 275
Software Initiated Bus Reset........................................................................................... 275
Bus Reset Signaling ............................................................................................................... 275
Effects of Bus Reset........................................................................................................... 277
Topology Information Cleared ...................................................................................... 277
PHY Register Changes .................................................................................................... 278
Port Bias and Connected Bits .................................................................................. 278
Port_Event Bit............................................................................................................ 278
Initiate Bus Reset (IBR) and Initiate Short Bus Reset (ISBR) ....................................... 278
Physical_ID Field ...................................................................................................... 278
Gap_Count................................................................................................................. 278
CSR Register Changes ..................................................................................................... 279
1394-1995 and Reset Runaway ............................................................................................. 279
Problem One: The Reset Storm ...................................................................................... 279
The 1394a Solution: Debounce Port Status Signal....................................................... 280
Problem Two: Recognition of Connection Change Not Symmetric............................. 282
The Solution: Slow Node Accepts Fast Node’s Reset Signaling .................................... 282
Problem Three: Reset Signaled During Packet Transmission........................................ 282
1394a Solution: Arbitrated Bus Reset ............................................................................. 283
## Chapter 15: Tree Identification

**Overview** ....................................................................................................................... 285  
**Tree ID Signaling** .............................................................................................................. 286  
**The Tree ID Process** ........................................................................................................... 286  
  - Leaf Nodes Try to Find Their Parents ........................................................................... 287  
  - Parents Identify Their Children ..................................................................................... 288  
**Three Example Scenarios** ...................................................................................................... 290  
**Scenario One** .......................................................................................................................... 290  
  - Leaf Nodes Signal Parent Notify .................................................................................. 290  
  - Branch Nodes Locate Their Parents .............................................................................. 292  
**Scenario Two** ................................................................................................................... 293  
  - Leaf Nodes Locate Their Parents ................................................................................... 294  
  - Root Contention ............................................................................................................. 295  
**Scenario Three** .................................................................................................................. 298  
  - Force Root Delay ........................................................................................................... 299  
  - Leaf Nodes Attempt to Locate Their Parents ............................................................... 300  
  - Branch Nodes Attempt to Locate Their Parents .......................................................... 300  
**Looped Topology Detection** ................................................................................................. 304

## Chapter 16: Self Identification

**Overview** ....................................................................................................................... 305  
**Self-Identification Signaling** ............................................................................................... 306  
**Physical ID Selection** ............................................................................................................ 306  
  - First Physical ID is Assigned ....................................................................................... 306  
  - Self-ID Count ................................................................................................................. 307  
  - Branch Nodes Signal Arbitration Grant & Data Prefix .............................................. 308  
  - Node A Broadcasts Its Self-ID Packet .......................................................................... 310  
  - Node A Signals Self-Identification Done ..................................................................... 311  
  - Nodes Exchange Speed Information ............................................................................ 312  
**Second and Subsequent Physical ID Assignment** ................................................................ 313  
  - Second Self-ID Assignment ......................................................................................... 314  
  - Third Physical ID Assignment ....................................................................................... 316  
  - Fourth Physical ID Assignment ..................................................................................... 318  
  - Fifth Physical ID Assignment ......................................................................................... 320  
  - Final Physical ID Always Belongs to Root Node ......................................................... 322  
**Self-ID Packets** ....................................................................................................................... 323  
  - Self-ID Packet Zero ........................................................................................................ 323  
  - Self-ID Packets One and Two (1394a) ......................................................................... 325  
  - Who Uses the Self-ID Packet Information .................................................................... 326
## Contents

### Part Four
**Serial Bus Management**

## Chapter 17: Cycle Master

- **Overview** ................................................................. 329
- **Determining and Enabling the Cycle Master** ................. 329
- **Cycle Start Packet** .................................................. 330

## Chapter 18: Isochronous Resource Manager

- **Overview** .................................................................. 333
- **Determining the Isochronous Resource Manager** ........... 334
- **Minimum Requirements of Isochronous Resource Managers** 335
- **Enabling the Cycle Master** ........................................ 335
- **Resource Allocation Registers** ..................................... 336
  - Channel Allocation ....................................................... 337
  - Channels Available Register Format ............................. 337
  - Accessing the Channels Available Register .................. 337
  - Bus Bandwidth Allocation ............................................ 339
  - Bandwidth Available Register Format .......................... 340
  - Accessing the Bandwidth Available Register .............. 340
  - Bus Bandwidth Set-Aside for Asynchronous Transactions .... 341
  - Reallocation of Isochronous Resources ...................... 342
- **Power Management** .................................................... 342

## Chapter 19: Bus Manager

- **Overview** ................................................................. 343
- **Determining the Bus Manager** .................................... 344
- **Power Management** .................................................... 345
  - Power Management by Bus Manager Node .................... 345
  - Power Management by IRM Node ................................. 346
- **The Topology Map** ..................................................... 346
  - Accessing the Topology Map .................................... 347
  - Gap Count Optimization ............................................ 348
- **The Speed Map** .......................................................... 349
  - Accessing the Speed Map .......................................... 349
- **Bus Bandwidth Set-Aside** .......................................... 350
## Contents

### Chapter 20: Bus Management Services

**Overview** .......................................................................................................................... 351

**Serial Bus Control Requests** ................................................................................................ 353

- Bus Reset Control Request ............................................................................................. 353
- Initialize Control Request .............................................................................................. 354
- Link-On Control Request ............................................................................................... 354
- Present Status .................................................................................................................... 354
- PHY Configuration Request ........................................................................................... 354
- Set Force Root and Set Gap Count ................................................................................. 355
- Extended PHY Packets .................................................................................................. 355

**Serial Bus Control Confirmations** ...................................................................................... 356

**Serial Bus Event Indication** ................................................................................................ 357

### Part Five

#### Registers & ROM

### Chapter 21: CSR Architecture

**Overview** .......................................................................................................................... 361

**Core Registers** ................................................................................................................ 362

- Effect of Reset on the CSRs ............................................................................................. 364
- State Register (State_Clear & State_Set) ........................................................................... 364
  - State_Clear Register .................................................................................................. 365
  - State_Set Register ................................................................................................... 366
- Bus Depend Field .......................................................................................................... 366
- Cycle Master Enable ...................................................................................................... 367
- Abdicate ......................................................................................................................... 369
- Node_IDS Register ........................................................................................................... 369
- Reset_Start Register ..................................................................................................... 373
- Indirect_Address and Indirect_Data Registers ................................................................ 373
- Split_Timeout Register .................................................................................................. 373
- Argument, Test_Start, and Test_Status Registers ........................................................ 375
- Units_Base, Units_Bound, Memory_Base, and Memory_Bound Registers .................... 375
- Interrupt_Target and Interrupt_Mask Registers ........................................................... 375
- Clock_Value, Clock_Tick_Period, Clock_Strobe_Arrived, and Clock_Info Registers ........ 376
- Message_Request & Message_Response Registers ......................................................... 376

**Serial Bus Dependent Registers** ......................................................................................... 376

- Cycle_Time & Bus_Time Registers .................................................................................... 377
- Power_Fail_Imminent & Power_Source Registers ............................................................ 380
- Busy_Timeout Register .................................................................................................... 382
Part Six
Power Management

Chapter 24: Introduction to Power Management
Overview ............................................................................................................................ 427
Review of 1394-1995 Power-Related Issues ................................................................. 428
Goals of the 1394a Power Extensions ............................................................................ 429

Chapter 25: Cable Power Distribution
Power Distribution ........................................................................................................... 431
Power Class Codes ....................................................................................................... 432
Power Providers .......................................................................................................... 433
  Power Provider Classes ............................................................................................ 434
  Alternate Power Providers ..................................................................................... 436
    Maximum Voltage <20vdc .................................................................................. 436
    Maximum Voltage >20vdc .................................................................................. 436
Power Consumer ......................................................................................................... 437
Self-Powered Nodes (Non Power Providers) ............................................................... 438
  Self-Powered Class Zero Nodes ............................................................................. 439
  Self-Powered Class Four Nodes ............................................................................. 440
    Two port node — no cable power used ............................................................... 440
    Three ports or more — no cable power used .................................................... 441
    Two ports or more — cable power used when power is lost ............................. 441
    Two ports or more — cable-powered PHY ....................................................... 442
Local Power Down Summary ....................................................................................... 443
## Chapter 26: Suspend & Resume

**Overview** ....................................................................................................................... 445

**Suspending a Port** ........................................................................................................... 448
- Suspending Via the Suspend Command Packet .............................................. 449
- Suspending Via RX_SUSPEND ........................................................................... 451
- The BIAS Handshake ............................................................................................. 451
- Suspending Via Port Disable .................................................................................. 451
  - Disable Via Disable Port Command Packet (local or remote) .................. 452
- Port Suspend Via Unexpected Loss of Bias .......................................................... 453

**Resuming Full Operation** ............................................................................................. 454
- Resuming Via Resume Packet .............................................................................. 454
- Resuming Via Resume Port Command Packet ................................................... 454
- Resuming Via Port Events ..................................................................................... 455

## Chapter 27: Power State Management

**Power Management** ........................................................................................................ 459

**Power States** .................................................................................................................... 460
- Node Power States ................................................................................................. 460
  - Node Power State Zero (N0) ............................................................................ 461
  - Node Power State One (N1) .............................................................................. 461
  - Node Power State Two (N2) .............................................................................. 461
  - Node Power State Three (N3) ........................................................................... 461
- Unit Power States ...................................................................................................... 461

**New CSRs** ...................................................................................................................... 463
- Node Power Control Register .............................................................................. 464
- Notification Address Register .............................................................................. 466
- Cable Power Source State Register .................................................................... 466
- Cable Power Source Control Register ................................................................ 467
- Power Change Register ......................................................................................... 468
- Unit Power State Register ..................................................................................... 469
- Unit Power Control Register ................................................................................ 470
- Battery State Register ........................................................................................... 471

**New ROM Entries** ......................................................................................................... 472
- Node Power Directory Entry ................................................................................. 473
- Node Power Level Entry ......................................................................................... 474
- Cable Power Source Level Entry ......................................................................... 475
- Node Power Management Entry .......................................................................... 475
- Battery Group Entry (Node) .................................................................................. 476
- Battery State Entry ................................................................................................ 476
- Unit Power Directory Entry ................................................................................... 477
1 Why FireWire?

This Chapter

This chapter provides a brief history of FireWire (IEEE 1394). It also discusses the need for FireWire and reviews the applications for which it is well suited.

The Next Chapter

The next chapter describes the primary features of the FireWire serial bus implementation. The chapter also reviews the IEEE 1394 standards (IEEE 1394-1995 & IEEE 1394.A) and IEEE ISO/IEC 13213 (ANSI/IEEE 1212) standard that the FireWire serial bus is based upon.

Overview

Development of FireWire began in the mid 1980s by Apple Computer. In fact, the term FireWire is a registered trademark of Apple Computer Corporation. As other manufacturers gained interest in FireWire, a working committee was formed to create a formal standard on the architecture. The resulting specification was submitted to IEEE and IEEE 1394-1995 was adopted.

Motivations Behind FireWire Development

FireWire provides a serial bus interconnect that allows a wide range of high performance devices to be attached. A variety of issues led to the development of FireWire. The primary characteristics of this serial bus include:

- Ease of use
- Low cost device implementations
- High speed application support
- Scalable performance
- Support for isochronous applications
- Huge amount of memory mapped address space supported (16 exabytes)
- Operation independent of host system
FireWire System Architecture

Inexpensive Alternate to Parallel Buses

The IEEE 1394 serial bus provides an alternative to more expensive parallel bus designs. Benefits of the serial bus over most parallel bus implementations are listed below.

- Reduced cost compared with many parallel bus implementations.
- Peripherals in current personal computer systems reside on a variety of buses (e.g. PCI and ISA buses). Communication between such devices can be problematic due to bus protocol and speed differences, thus slowing overall performance. FireWire provides an opportunity to locate a wide variety of peripheral devices that connect to the same serial bus, resulting in performance gains. Up to 63 nodes can be attached to a single serial bus.
- Many parallel buses are confined to a small physical area; however, serial bus has much greater flexibility (4.5 meters between devices).
- FireWire supports direct attachment of remote peripherals.
- The 1394 bus can be implemented in conjunction with the parallel bus to provide fault tolerance.

Plug and Play Support

Devices attached to the IEEE 1394 serial bus support automatic configuration. Unlike USB devices, each 1394 node that attaches to the bus automatically participates in the configuration process without intervention from the host system. Each time a new device is added to or removed from the bus, the 1394 bus is re-enumerated. This occurs whether or not the bus is attached to a host system.

Eliminate Host Processor/Memory Bottleneck

Like any bus that supports bus mastering, the 1394 bus has the ability to increase overall system performance. In a PC environment the 1394 bus can reduce traffic across PCI and reduce accesses to the memory subsystem. This can be accomplished by locating devices on the 1394 bus that communicate with each other frequently. This eliminates the need for the processor and memory subsystems to be involved in the transfer of data between devices.
Chapter 1: Why FireWire?

High Speed Bus with Scalable Performance

Many peripheral devices such as hard drives and video cameras require high throughput. The 1394 bus accommodates these types of devices with a 400Mb/s transfer rate. This yields a theoretical throughput of 50MB/s in contrast to the throughput of ISA (8MB/s) and PCI (132MB/s). The 1394 serial bus provides scalable performance by supporting transfer rates of 400Mb/s, 200Mb/s, and 100Mb/s.

Support for Isochronous Applications

The serial bus supports isochronous transfers to support applications such as audio and video which require constant transfer rates. The isochronous transfer support reduces the amount of buffering required by isochronous applications, thereby reducing cost.

BackPlane and Cable Environments

1394 supports both a backplane and cable implementation, permitting flexibility of implementation. The backplane environment provides the ability of establishing a redundant serial bus communications channel in conjunction with a parallel bus implementation. The cable environment allows the remote attachment of peripheral devices with the possibility of supporting peripherals spread over a distance of greater than 250 meters. The capability makes the serial bus an attractive option for small network applications.

Bus Bridge

The huge amount of memory address space supported, high transfer rates, and low costs make the 1394 bus an attractive means of bridging between different host systems and between multiple serial bus implementations.

- Serial bus implementations can be used to bridge other buses together. The serial bus provides the ability to bridge between host systems of varying sizes and types, including PCs, mini-computers, and mainframes.
- A single serial bus supports 63 nodes but can support up to 1024 serial buses, making the total number of nodes supported at nearly 64k.
FireWire System Architecture

1394 Applications

The scalable performance and support for both asynchronous and isochronous transfers makes FireWire an alternative for connecting a wide variety of peripherals including:

- Mass storage
- Video teleconferencing
- Video production
- Small networks
- High speed printers
- Entertainment equipment
- Set top box

IEEE 1394 Refinements

Early implementations based on different interpretations of the 1995 release of the specification resulted in some interoperability problems between different vendor parts. A supplement to the 1394-1995 specification is referred to as the 1394a supplement, and is designed to eliminate these problems. In addition to clarifying portions of the 1394-1995 specification, the 1394a supplement fixes problems, specifies enhancements that are designed to improve performance, and adds new functionality. This book covers the 1394a supplement (2.0 draft version) as it existed at the time of writing.

Power management support and a specification for designing 1394 bridges were also in development at the time of this writing. Portions of the preliminary Power Management specification are included in this text, while the state of the bridge specification was not mature enough to be included.

Yet another version of 1394 being developed is called the IEEE 1394.B specification. This specification defines even higher throughput including 800Mb/s, 1.6Gb/s, and 3.2Gb/s. This specification is being designed for backward compatibility to 1394-1995 and 1394a.
Chapter 1: Why FireWire?

Primary Features

The primary features of FireWire’s cable environment are summarized in Table 1-1 on page 17. The next chapter provides a more detailed overview of the FireWire architecture.

Table 1-1: FireWire Key Features

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalable Performance</td>
<td>Speeds of 100, 200, and 400 Mb/s supported.</td>
</tr>
<tr>
<td>Hot Insertion &amp; Removal</td>
<td>Devices can be attached or removed from the bus dynamically without powering the system down.</td>
</tr>
<tr>
<td>Plug and Play</td>
<td>Each time a device is attached or detached the bus is re-enumerated. Nodes on the bus are to a large degree self-configuring, and configuration does not require intervention from a host system (such as a PC).</td>
</tr>
<tr>
<td>Support for two types of transactions</td>
<td>Support for isochronous and asynchronous transfers.</td>
</tr>
<tr>
<td>Layered hardware and software model</td>
<td>Communications based on a transaction layer, link layer, and physical layer protocols.</td>
</tr>
<tr>
<td>Support for 64 nodes</td>
<td>Supports 64 node addresses (0-63) on a single serial bus implementation. Node address 63 is used as a broadcast address that all nodes recognize, permitting attachment of up to 63 physical nodes on the bus.</td>
</tr>
<tr>
<td>Address space of 16 petabytes per bus</td>
<td>Each of the 64 nodes has 256TB of address space, making the total address space of 16 petabytes.</td>
</tr>
<tr>
<td>Support for 1024 buses</td>
<td>The CSR architecture supports up to 1024 buses for a total address space of 16 exabytes.</td>
</tr>
<tr>
<td>Peer-to-Peer transfer support</td>
<td>Serial bus devices have the ability to perform transactions between themselves, without the intervention of a host CPU.</td>
</tr>
</tbody>
</table>
Overview of the IEEE 1394 Architecture

The Previous Chapter

The previous chapter discussed the need for FireWire and reviewed the applications for which it is well suited.

This Chapter

This chapter describes the primary features of the FireWire serial bus implementation. The chapter also overviews the IEEE 1394 standards (IEEE 1394-1995 & IEEE 1394a) and ISO/IEC 13213 (ANSI/IEEE 1212) standard that the FireWire serial bus is based upon.

The Next Chapter

The next chapter provides an overview of the IEEE 1394 communications model. It defines the basic transfer types and introduces the communication layers defined by the serial bus specification.

IEEE 1394 Overview

The IEEE 1394 specification defines the serial bus architecture known as FireWire. Originated by Apple Computer, FireWire is based on the internationally adopted ISO/IEC 13213 (ANSI/IEEE 1212) specification. This specification, formally named “Information technology—Microprocessor systems—Control and Status Registers (CSR) Architecture for microcomputer buses,” defines a common set of core features that can be implemented by a variety of buses. IEEE 1394 defines serial bus specific extensions to the CSR Architecture.
FireWire System Architecture

IEEE 1394-1995 provides support for a backplane environment and a cable environment. This book focuses only on the cable environment.

Specifications and Related Documents

This book is based on a variety of specifications and documents, some of which are completed and approved, while others are in varying stages of completion and approval. To distinguish information based on approved specifications versus information based on pre-approved specifications and documents, a symbol is used to alert the reader to pre-approved references that might otherwise not be obvious. The following bulleted list includes the specifications and documents used as references when writing this book. The symbol previously mentioned is used to highlight those specifications and documents that are not approved at the time of this writing.

The following documents were used during the development of this book:

  - IEEE 1394a Supplement
  - Power Distribution Specification
  - Power Management Specification
  - Suspend/Resume Specification
  - Open Host Controller Interface (OHCI) for 1394 Specification
- Device Bay Specification
- 1394.1 Bridge Specification
- IEEE 1394.B Specification

The author strongly urges the reader to obtain the latest (and hopefully approved) versions of these specifications. Also please check MindShare’s web site (www.mindshare.com) to check for errata, clarifications, and additions to this book. Due to the evolving nature of this topic, many changes are inevitable. Some of these specifications may be available on the IEEE 1394 Trade Association web site at: www.firewire.org.

IEEE 1394-1995 and the IEEE 1394a Supplement

The IEEE 1394 specification was released in 1995, hence the name IEEE 1394-1995. Different interpretations of the 1995 specification have led to interoperability problems. To clarify the specification a supplement to the 1995 specifica-
Chapter 2: Overview of the IEEE 1394 Architecture

A higher speed 1394 serial bus called the “B” version was also being developed, when this book was being written. This specification will define serial bus extensions for running the serial bus at speeds into the gigabit per second range. This specification is intended to be backwardly compatible with the 1394-1995 and 1394a implementations. Note that another solution has been proposed that also increases the serial bus speed, and is known as the 1394.2 version. This proposed solution, however, is not backwardly compatible with earlier 1394 versions, causing considerable opposition.

Unit Architecture Specifications

A wide variety of functional devices (e.g. hard drives, video cameras, and CD-ROMs) can be implemented as 1394 nodes. Functional devices are termed units by the 1394 specification. Certain types of devices may have related specifications called “unit architectures” that define implementation details such as protocols, ROM entries, control and status registers, etc. Two specifications of this type are:

- Serial Bus Protocol 2 (SBP2) Architecture — used for SCSI-based mass storage functions.
- A/V unit Architecture — used for audio/visual functions.

Check the 1394 Trade Association web site (www.firewire.org) for information regarding Unit Architecture documentation.
Figure 2-1: PC with IEEE 1394 Bus Attached to the PCI Bus.
IEEE 1394 Topology

Figure 2-1 on page 22 represents a typical PC that incorporates an IEEE 1394 serial bus attached to the PCI bus. A PCI to 1394 bridge (Open Host Controller Interface, or OHCI) interfaces the computer to the serial bus. The serial bus allows attachment of high speed peripheral devices that would otherwise require a relatively expensive bus solution such as PCI or SCSI. As shown in Figure 2-1, a wide variety of peripheral devices can be attached and supported.

Multiport Nodes and Repeaters

1394 nodes may have one or more ports. A single port node discontinues the bus along a given branch of the bus, whereas nodes with two or more ports allow continuation of the bus, as illustrated in Figure 2-2 on page 23. Nodes with multiple ports permit the bus topology to be extended. Note that the signaling environment is point-to-point. That is, when a multiport node receives a packet, it is detected, received, resynchronized to the repeaters local clock and retransmitted over the other node ports.

Configuration

Configuration is performed dynamically as new devices are attached and/or removed from the bus. The configuration process does not require intervention from the computer system.
This chapter provides an overview of the serial bus communications model. It defines the basic transfer types and introduces the communication layers defined by the specification.

The Next Chapter

The next chapter describes the services defined by the specification that are used to pass parameters between layers during the execution of each transaction.

Overview

Since the IEEE 1394 serial bus supports peer-to-peer transactions, arbitration must be performed to determine which node will obtain ownership of the serial bus. This arbitration mechanism supports a fairness algorithm that ensures that all transfers obtain fair access to the bus. Serial bus arbitration is discussed in detail in Chapter 7.

The serial bus supports two data transfer types:

- asynchronous transfers that do not require delivery at a constant data rate. Asynchronous transfers target a particular node based on a unique address. These transfers do not require a constant bus bandwidth and therefore do
FireWire System Architecture

- not need regular use of the bus, but must get fair access over time.
- isochronous transfers that require data delivery at constant intervals. These transfers define a channel number rather than a unique address, permitting the isochronous data stream to be broadcast to one or more nodes responding to the channel number. These transfers require regular bus access and therefore have higher bus priority than asynchronous transfers.

Serial bus data transactions take place via a series of data and information packet transmissions. Each transaction is initiated by a “requester” and the request is received by a target device, called a “responder.”

Asynchronous transactions require a response from the target node, which results in an additional transaction. The responding node either accepts or returns data as illustrated in Figure 3-1.

*Figure 3-1: Request/Response Protocol*

This basic communications model is defined by the CSR architecture. The serial bus provides further granularity in the transaction process, which includes verification of packet delivery, three protocol layers, and related services that perform specific functions during the process of transferring data between the requester and responder. These layers are described later in this chapter.
Chapter 3: Communications Model

Isochronous transactions complete following the request as illustrated in Figure 3-2. Note that rather than an address, isochronous transactions use a channel number to identify target nodes. Additionally no response is returned from the target node.

Figure 3-2: Isochronous Transaction that Consists Only of a Request Transaction

The actual transfer of data across the cable is done serially using data-strobe encoding (See page 122). Data can be transmitted at one of three speeds:

- 100 Mb/s (98.304 Mb/s)
- 200 Mb/s (196.608 Mb/s)
- 400 Mb/s (393.216 Mb/s)

Prior to transferring data, the transmitting node must obtain ownership of the 1394 bus via an arbitration mechanism. This ensures that only one node at a time is transmitting data over the wire.

Transfer Types

As illustrated in Figure 3-3, a mix of isochronous and asynchronous transactions may be performed across the serial bus by sharing the overall bus bandwidth. Notice that bus bandwidth allocation is based on 125µs intervals, called cycles. Details regarding these transaction types and their bandwidth allocation are discussed below.
Asynchronous transfers target a particular node by using an explicit 64-bit address. Asynchronous transfers (collectively) are guaranteed 20% (minimum) of the overall bus bandwidth. Thus, the amount of data transferred depends on the transmission speed. A given node is not guaranteed any particular bus bandwidth, but rather is guaranteed fair access to the bus via a fairness interval, in which each node wishing to perform an asynchronous transaction gets access to the bus exactly one time during a single fairness interval. Maximum packet size for asynchronous transfers is limited as specified in Table 3-1. Note that higher cable speeds have been specified by the 1394a supplement, but the maximum speed and maximum data payload supported remains at 400Mb/s.

Asynchronous transfers also verify data delivery via CRC checks and response codes, and in the event that errors occur during transmission, retries may be attempted under software control.
Chapter 3: Communications Model

Table 3-1: Maximum Data Block Size for Asynchronous Transfers

<table>
<thead>
<tr>
<th>Cable Speed</th>
<th>Maximum Data Payload Size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100Mb/s</td>
<td>512</td>
</tr>
<tr>
<td>200Mb/s</td>
<td>1024</td>
</tr>
<tr>
<td>400Mb/s</td>
<td>2048</td>
</tr>
<tr>
<td>800Mb/s</td>
<td>4096</td>
</tr>
<tr>
<td>1.6Gb/s</td>
<td>8192</td>
</tr>
<tr>
<td>3.2Gb/s</td>
<td>16384</td>
</tr>
</tbody>
</table>

Isochronous

Isochronous transfers target one or more (multi-cast transactions) devices based on a 6-bit channel number associated with the transfer. Channel numbers are used by the isochronous listeners to access a memory buffer within the application layer. This memory buffer may or may not reside within the node’s 256TB of address space.

Each isochronous application must also obtain the necessary bus bandwidth that it requires for its transfer. To ensure that sufficient bus bandwidth is available, applications wishing to perform isochronous transfers must request the needed bandwidth from the isochronous resource manager node. Bus bandwidth is allocated on a per cycle basis.

Once bus bandwidth has been acquired for an isochronous transfer, that channel receive a guaranteed time-slice during each 125µs cycle. Up to 80% (100µs) of each bus cycle can be allocated to isochronous transfers. The maximum packet size supported for a given isochronous transfer is limited to the available bus bandwidth, and must not exceed the maximum packet size specified in Table 3-2 on page 42. The maximum packet size limit for isochronous transactions has been added by the 1394a specification. The isochronous bus bandwidth available is maintained by the isochronous resource manager node. Each node wishing to perform isochronous transfers must request its desired bus bandwidth from the isochronous resource manager based of the number of desired allocation units. The maximum packet size may be limited by the
Communications Services

The Previous Chapter

The previous chapter provided an overview of the serial bus communications model. It defined the basic transfer types and introduced the communication layers defined by the specification.

This Chapter

This chapter describes the services defined by the specification that are used to pass parameters between layers during the execution of each transaction.

The Next Chapter

The next chapter discusses the cable characteristics and connectors used by the IEEE 1394 cable environment. It also discusses the Device Bay implementation being specified in PC environments.

Overview

The IEEE 1394 specification defines services that are used to pass parameters between each layer within the communications model. These services are used to initiate transactions or to respond to a transaction that has been received. The following sections describe the services used when performing asynchronous and isochronous transactions.
Anatomy of Asynchronous Transactions

Three primary types of asynchronous transactions are defined by the 1394 specification:

- reads
- writes
- lock

The following discussion reviews each of these transaction types and defines the protocol layers and services calls used to initiate and respond to asynchronous transactions. It also defines the packet types used and details the possible responses.

The following sections describe the steps involved in completing asynchronous transactions. The three basic asynchronous transaction types are described from initiation to response completion with the role of each layer in the protocol detailed.

The Request Subaction

The request subaction involves sending the request phase of an asynchronous transaction to a target node. Both the request and the response agent are involved in the request subaction as discussed below. The specification defines the protocol layers used during the request phase of an asynchronous transaction. The services used in performing a request subaction are listed in Table 4-1. These services are listed in order of reoccurrence during successful request subaction transmission. Note that the shaded entries indicate actions that take place within the responding node. These services are represented graphically in Figure 4-1 on page 68.

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Direction of Communication</th>
<th>Purpose of Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Data Request</td>
<td>From the Application</td>
<td>Causes transaction layer to initiate an asynchronous transaction.</td>
</tr>
<tr>
<td>Link Data Request</td>
<td>From Transaction Layer</td>
<td>Causes link layer to initiate an asynchronous transaction.</td>
</tr>
</tbody>
</table>
Table 4-1: Service Used During Asynchronous Request Subactions (Continued)

<table>
<thead>
<tr>
<th>Service Name</th>
<th>Direction of Communication</th>
<th>Purpose of Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>PHY Arbitration Request</td>
<td>From Link Layer</td>
<td>Causes the PHY to arbitrate for control of the serial bus.</td>
</tr>
<tr>
<td>PHY Arbitration Confirmation</td>
<td>From PHY</td>
<td>Reports results of arbitration request back to link.</td>
</tr>
<tr>
<td>PHY Clock Indication</td>
<td>From PHY</td>
<td>Following successful arbitration, the PHY notifies the link that it is ready to accept clocked data.</td>
</tr>
<tr>
<td>PHY Data Request</td>
<td>From Link Layer</td>
<td>Controls clocked transmission of the request packet onto the serial bus.</td>
</tr>
<tr>
<td>PHY Data Indication</td>
<td>To Link Layer</td>
<td>Notifies link of the receipt of data bits of the packet.</td>
</tr>
<tr>
<td>Link Data Indication</td>
<td>To Transaction Layer</td>
<td>Indicates the reception of a transaction request.</td>
</tr>
<tr>
<td>Transaction Data Indication</td>
<td>To Application</td>
<td>Indicates the reception of a transaction request.</td>
</tr>
<tr>
<td>Link Data Response</td>
<td>From Transaction Layer</td>
<td>Initiates return of acknowledge packet to the requesting node.</td>
</tr>
<tr>
<td>PHY Data Request</td>
<td>From Link Layer</td>
<td>Controls clocked transmission of the acknowledge packet onto the serial bus.</td>
</tr>
<tr>
<td>PHY Data Indication</td>
<td>To Link Layer</td>
<td>Notifies link of the receipt of the acknowledgment packet.</td>
</tr>
<tr>
<td>Link Data Confirmation</td>
<td>To Transaction Layer</td>
<td>Notifies transaction layer whether the request was successfully received.</td>
</tr>
</tbody>
</table>
Figure 4-1: Example Asynchronous Read Transaction

<table>
<thead>
<tr>
<th>Physical Layer</th>
<th>Physical Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Port 0</td>
<td>Port 1</td>
</tr>
</tbody>
</table>

- **Physical Data Requests**
- **Physical Data Indications**

<table>
<thead>
<tr>
<th>Link Layer</th>
<th>Link Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Link Data Request**
- **Link Data Confirmation**

<table>
<thead>
<tr>
<th>Transaction Layer</th>
<th>Transaction Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Transaction Data Request**
- **Transaction Data Confirmation**

<table>
<thead>
<tr>
<th>Application</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Asynchronous Transfer Interface**

<table>
<thead>
<tr>
<th>Node A (Requester)</th>
<th>Node B (Responder)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Read Request**
- **Read Response**
- **Acknowledgment**

Serial Bus

Port 1
Chapter 4: Communications Services

Initiating the Transaction (The Request)

**Transaction Layer.** Applications initiate asynchronous transactions via the “transaction data request” service. Transaction layer services can be thought of as calls to low level routines that insulate the application programmer from the programming interface associated with the link layer controller chip. The transaction layer also provides verification of packet delivery and initiates the acknowledge packet. The “transaction data request” service, communicates the following parameters:

- Transaction Type Code
  - Write
  - Read
  - Lock
- Extended transaction code (only defined for lock transaction)
- Destination Address
- Data Length
- Data (Data to be transferred during a write or lock transaction)
- Speed of transmission

**The Link Layer.** The transaction data request calls a link layer service routine that passes the transaction parameters specified by the application onto the link layer controller in the form that it understands. In addition, the retry code and transaction label parameters are added as the data request is passed onto the link layer controller. The actual mechanism for sending commands to the link layer controller is not defined by the specification. Parameters passed by the “link data request” service include:

- Destination Address
- Transaction Type Code
  - Write
  - Read
  - Lock
- Extended transaction code (only defined for lock transaction)
- Data Length
- Data (Data to be transferred during a write or lock transaction)
- Speed of transmission
- Retry code
- Transaction Label
Cables & Connectors

The Previous Chapter

The previous chapter described the services defined by the specification that are used to pass parameters between layers during the execution of each transaction.

This Chapter

This chapter discusses the cable characteristics and connectors used by the IEEE 1394 cable environment. It also discusses the Device Bay implementation being specified in PC environments.

The Next Chapter

Next, the serial bus signaling environment is discussed. This includes recognition of device attachment and removal, arbitration signaling, speed signaling, and data/strobe signaling.

Cable and Connector Types

Two types of cables are supported by 1394. The original IEEE 1394-1995 specification defines a single 6-pin connector type and cable. The connectors are identical at both ends of the cable and can be plugged in either direction, between nodes.

The 1394a supplement defines an alternate 4-pin connector and cable that eliminates the power pins. Cables using this connector may have a 4-pin connector on one end of the cable and a 6-pin connector on the other end, or may have 4-pin connectors on each end. The specification places limits on the types of devices allowed to use 4-pin connectors.
FireWire System Architecture

6-pin Connector (1394-1995)

The original 1394-1995 specification defines a 6-pin plug and socket that is illustrated in Figure 5-1. The contact signal assignments are listed in Table 5-1 on page 87. Cable assemblies based on the 1995 version of the specification have mechanically-identical plugs at each end of the cable and all 1394 devices employed the standard 6-pin socket.

Figure 5-1: 6-Pin Plug and Socket
The socket dimensions are relatively small (11.3mm X 6.2mm) when compared to standard connectors used by many computer peripheral devices. The socket consists of a shell and contact wafer. The plug body fits into the socket shell and the contacts within the plug body slide over the socket’s contact wafer as the plug is inserted.

**Make First/Break Last Power Pins**

The 6-pin socket has longer contact power and ground contact pins. This ensures that the power pins make contact prior to the data pair pins when the plug is inserted into the socket; and conversely, when a plug is removed the data pins break contact prior to the power pins. The separation between the power and data pins is specified to be a minimum of 0.8mm.

**Optional 4-pin Connector (1394a supplement)**

The 4-pin connector is defined by the 1394a supplement for use in 1394 applications where the standard 6-pin connector is too large. Figure 5-2 illustrates the 4-pin plug and socket. The connector was originally designed by Sony and included in their video cameras. The 1394a supplement adopted the Sony design. The specification defines two categories of 1394 devices that may benefit from a smaller connector:

- Battery operated devices — Since these nodes do not draw power from the cable, a less expensive cable and connector are possible and desirable. Fur-
Furthermore, the power conductors can be a source of unwanted analog noise, which is a major concern for applications that include audio.

- Hand-held devices — The standard connector may be relatively bulky when implemented into small hand-held devices such as video camcorders.

*Figure 5-2: 4-Pin Plug and Socket*

Positive Retention

Both connector types employ positive retention via a detent. Applying sufficient force releases the plug when removing the cable. The specification permits stronger retention features to be implemented. However, these additional retention features must not interfere with the ability to mate the plug or connector using the standard detent retention mechanism.
Chapter 5: Cables & Connectors

Cable Characteristics

Cable electrical characteristics are the same for the 4-conductor and 6-conductor cable, with the exception that the 4-conductor cable does not include the power wires. Standard electrical characteristics of the cables have the following parameters and characteristics. Test and measurement procedures are described in the specification.

- Suggested maximum cable length = 4.5 meters (with signal velocity = 5.05ns/meter)
- 110 ohms characteristic impedance — differential mode
- 33 ohms characteristic impedance — common mode
- Signal velocity equal to or less than 5.05ns/meter
- Signal pair attenuation:
  - 100MHz = <2.3dB
  - 200MHz = <3.2dB
  - 400MHz = <5.8dB
- Relative propagation skew ≤ 400ps @ 100MHz and ≤ 100ps @ 400MHz (the difference between the differential mode propagation delay of the two twisted pair conductors that must be measured in the frequency domain).
- TPA to TPB Crosstalk ≤ -26 dB (within 1MHz to 500MHz range).

6-Conductor Cables

Figure 5-3 illustrates the cross-section of a 6-conductor cable including the wires and insulation required.

*Figure 5-3: Cross-section of 6-Conductor Cable*
The Electrical Interface

The Previous Chapter
The previous chapter discussed the cable characteristics and connectors used by the IEEE 1394 cable environment. It also discussed the Device Bay implementation being specified in PC environments.

This Chapter
This chapter details the serial bus signaling environment. This includes recognition of device attachment and removal, arbitration signaling, speed signaling, and data/strobe signaling.

The Next Chapter
The next chapter discusses the arbitration process. It defines the various types of arbitration including isochronous and asynchronous arbitration, as well as the newer arbitration types defined by the 1394a supplement.

Overview
The IEEE 1394 serial bus employs two twisted pairs of signal wires (Twisted Pair A, or TPA and Twisted Pair B, or TPB). Additionally, a single pair of wires may be used to provide power for nodes. TPA and TPB provide both differential and common mode signaling to support the following functions.

- Recognition of device attachment/detachment
- Reset
- Arbitration
- Packet transmission
- Automatic configuration
- Speed signaling
FireWire System Architecture

The twisted pair signals are crosswired within the cable, such that TPA and TPB of one node connects respectively to TPB and TPA of the other. The individual twisted pair signals are referred to as TPA/TPA* and TPB/TPB*. Either node may initiate a transaction and therefore use identical signaling interfaces. The following sections discuss the functional aspects of the 1394 signaling environment.

Common Mode Signaling

Common mode signaling is used for the following functions:

• Device attachment/detachment detection
• Speed signaling
• Suspend/resume signaling

These signaling environments utilize DC signals to accomplish the required functionality. The characteristic impedance of the signal pairs is 33±6Ω. Since the signaling is based on DC signaling, there is no concern regarding unwanted reflections. Common mode values are specified as the average voltage on the twisted pair A or B (e.g. Average voltage of TPA and TPA*).

Differential Signaling

Differential signaling is used for the following functions:

• Reset
• Arbitration
• Configuration
• Packet transmission

Differential signaling can occur at speeds of 100, 200, or 400MHz. The goal of the 1394 differential signaling environment is to eliminate signal reflections from occurring over the cable. This is accomplished by terminating the differential pairs to obtain a reflection coefficient of zero. The characteristic impedance of each signal is 110±6Ω, therefore 110Ω termination resistors (two 55Ω resistors in series) are employed to eliminate reflections on each signal line. See “Differential Signal Specifications” on page 104 for details regarding the differential signaling environment.
Chapter 6: The Electrical Interface

Differential signaling has two major advantages that are used by the 1394 bus:

- Noise immunity
- Three signaling states: differential 1, differential 0, and Hi Z

The three signaling states are used to define a variety of bus conditions and are detailed later in this chapter.

Recognition of Device Attachment and Detachment

Recognition of whether a device is attached to a given port or not differs between the 1394-1995 specification and the 1394a supplement. Both mechanisms are described in the following sections.

IEEE 1394-1995 Device Attachment/Detachment

Each node provides an offset voltage on its TPA signal lines by driving a TpBias voltage in the range of 1.665v to 2.015v. This voltage is driven by a twisted pair bias voltage source driver as illustrated in Figure 6-1. On the other end of the cable, TpBias is detected by the attached node’s port status receiver via TPB. Accounting for signal attenuation across the wire, the receiver senses a voltage between 1.165v and 2.015v.
Note that both signaling pairs have the bias voltage permanently applied once another node is attached. Each node detects the bias voltage being applied by the node on the opposite end of the cable. A port status receiver continuously compares 0.8vdc reference voltage to the voltage on the cable. When a node is attached to a given port, the bias voltage causes the cable voltage to rise above 0.8vdc, thereby signifying the attachment of a node. Table 6-1 gives the threshold voltages for the port receiver. When a previously attached node is removed from the network, the bias voltage will be also removed, causing the port status receivers to detect node removal.
IEEE 1394a Device Attachment/Detachment

The port suspend feature introduced by the 1394a supplement permits port circuitry to enter a low power state. In this state only TpBias and port connect status monitoring takes place. A suspended port is required to remove TpBias to confirm that it has entered the suspended state. A 1394-1995 node connected to the suspended port would detect the removal of TpBias and assume that the node had been detached, when in fact it is still connected but suspended. To differentiate between a suspended port and a detached node, new circuitry has been added to 1394a compliant nodes.

When a port enters the suspend state it must activate its port connection detect circuit (new to 1394a). This circuit monitors the physical connection between suspended ports. Figure 6-2 illustrates the connection detect circuitry. In the absence of TpBias, the connection_detect receiver will recognize detachment of a node. The current source \( I_{\text{CD}} \) must not exceed 76\( \mu \)A to ensure that the Port_Status receiver of the attached node does not exceed 0.4 vdc. If the attached node is removed, the input voltage at the connection detect circuit will rise above 0.4 vdc because the current path to ground will have been removed. This permits the port to recognize that the previously attached node has been disconnected.

Note that the connection_detect receiver is only valid when TpBias has been disabled. TpBias is removed when a port is either suspended, disabled, or disconnected.

<table>
<thead>
<tr>
<th>Port_Status</th>
<th>Common Mode Input Signal State (vdc)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node Detached</td>
<td>TPB input ≤ 0.6v</td>
</tr>
<tr>
<td>Indeterminate</td>
<td>0.6v &lt; TPB input &lt; 1.0v</td>
</tr>
<tr>
<td>Node Attached</td>
<td>TPB input ≥ 1.0v</td>
</tr>
</tbody>
</table>

Note that the connection_detect receiver is only valid when TpBias has been disabled. TpBias is removed when a port is either suspended, disabled, or disconnected.
Arbitration

The Previous Chapter

In the previous chapter, the serial bus signaling environment was detailed. The chapter covered recognition of device attachment and removal, arbitration signaling, speed signaling, and data/strobe signaling.

This Chapter

This chapter details the arbitration process. It defines the various types of arbitration including isochronous and asynchronous arbitration, as well as the newer arbitration types defined by the 1394a supplement.

The Next Chapter

Asynchronous transactions exist in three basic forms: reads, writes, and locks. The next chapter details the asynchronous packets that are transmitted over the bus.

Overview

Arbitration is based on guaranteed bus bandwidth for isochronous channels and a fairness interval for asynchronous channels. Arbitration begins when a node recognizes a period of bus idle time, thereby indicating the end of the previous transmission. The period of bus idle time, or gap timing, varies between isochronous and asynchronous transactions as follows:

- isochronous gap — the period of bus idle time during isochronous data transmission that must be observed prior to the arbitration for the next isochronous transaction. The isochronous gap detection must be between 0.04µs and 0.05µs.
FireWire System Architecture

- subaction gap — the period of bus idle time during asynchronous data transmission that must be observed prior to arbitration starting for the next asynchronous transaction. This gap can be tuned so that arbitration can begin as early as possible without interfering with the normal completion of a subaction and its subsequent acknowledgment.

The first discussion in this chapter focuses on arbitration signaling because the arbitration signaling protocol is identical for both isochronous and asynchronous transactions. Next, the arbitration services are reviewed that are used by the link layer to request ownership of the bus via the physical layer. Since the arbitration process for asynchronous transactions is distinctly different from the arbitration process used for isochronous transactions, each topic is discussed separately. Finally, the two arbitration processes are discussed in light of the effect each has on the other. This occurs when a mix of isochronous and asynchronous transactions are being performed at the same time.

Arbitration Signaling

When any node on the bus wishes to perform a transaction it must arbitrate for use of the bus. Arbitration priority is based on which node requesting bus ownership receives grant from the root node. All nodes wishing to obtain bus ownership signal TX_REQUEST toward the root node. Any node that detects the arbitration request on one of its ports must forward the request on toward the root unless it is already signaling request either for itself or another node. Ultimately an arbitration request will reach the root node. The root then signals TX_GRANT to the first port on which it detects a RX_REQUEST. When RX_REQUEST from two nodes vying for control of the bus happen to reach the root at the same time, the root signals TX_GRANT on the requesting port that is designated with the lower port number.

Table 7-1 reviews the arbitration line states used during arbitration.
Figure 7-1 illustrates a community of nodes residing on the serial bus with two nodes attempting to win use of the bus. Nodes A and E both signal an arbitration request (TX_REQUEST) to their parent nodes. Node A’s TX_REQUEST is detected by node B, whose job it is to forward the request on to its parent. At the same time node E also signals TX_REQUEST to its parent (the root). Since Node E connects directly to the root node, its request reaches the root before Node E’s request. When the root detects the request, it recognizes that a node is requesting use of the bus.

This example highlights the natural priority that exists due to the topology of the serial bus. Since the root is the source of the arbitration grant, it has the highest priority, followed by the nodes that connect directly to the root (lowest numbered ports first), etc. Natural arbitration priority for a given node then is based on the distance from the root node relative to other nodes. Note however, that nodes may not send TX_REQUEST at the same time; thus, natural priority does not guarantee that a node closer to the root will necessarily win arbitration over one further away from the root.

Table 7-1: Arbitration Line States During Normal Arbitration

<table>
<thead>
<tr>
<th>Line State Name Transmitted</th>
<th>Line State Transmitted</th>
<th>Line State Name Received</th>
<th>Line State Received/decoded</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX_REQUEST (child node)</td>
<td>Z</td>
<td>0</td>
<td>RX_REQUEST (parent node)</td>
</tr>
<tr>
<td>TX_GRANT (parent node)</td>
<td>Z</td>
<td>0</td>
<td>RX_GRANT (child node)</td>
</tr>
<tr>
<td>TX_REQUEST removed by child node</td>
<td>Z Z</td>
<td>0 0</td>
<td>RX_REQUEST_CANCEL (parent node detects its own TX_GRANT)</td>
</tr>
<tr>
<td>TX_DATA_PREFIX (parent node)</td>
<td>0 0</td>
<td>RX_DATA_PREFIX (child node)</td>
<td>1 0</td>
</tr>
</tbody>
</table>
Upon detecting a request on its port number two, the root immediately returns TX_GRANT (to node E). The root also signals a DATA_PREFIX to all other ports (just port 1 in this case) to notify all nodes downstream from the root that it has granted the buses to a node and that a packet can be expected (See Figure 7-2).

When node E recognizes the TX_GRANT, it removes its request and begins packet transmission. Note that node C forwards the DATA_PREFIX that it receives from the root to all of its children. Node B detects this DATA_PREFIX from node C at the same time that it signals TX_REQUEST to node C. The DATA_PREFIX causes node B to stop signaling TX_REQUEST and to forward the DATA_PREFIX to its child port (node A). Node A detects the DATA_PREFIX and removes its TX_REQUEST, and recognizes that another node has won control of the bus.
Figure 7-3 illustrates the state of the bus once the arbitration has been completely resolved. For a review of arbitration signaling line states, see “Line States During Normal Arbitration” on page 112.

Arbitration Services

When a link wishes to transmit a packet it must first request the PHY to obtain ownership of the bus. The type of packet to be transmitted determines the type of request that the LINK will make. The link layer must use one of four arbitration services when requesting bus ownership:

- Fair arbitration service (used when transmitting an asynchronous packet)
- Priority arbitration service (used when transmitting a cycle start packet or an asynchronous packet of high priority)
Asynchronous Packets

The Previous Chapter
The previous chapter detailed the arbitration process. It defined the various types of arbitration including isochronous and asynchronous arbitration, as well as the newer arbitration types defined by the 1394a supplement.

This Chapter
Asynchronous transactions exist in three basic forms: reads, writes, and locks. This chapter details the packets that are transmitted over the bus during asynchronous transfers.

The Next Chapter
The next chapter discusses isochronous transactions. These transactions are scheduled so that they occur at 125µs intervals. The chapter discusses the role of the application, link, and PHY in initiating and performing isochronous transactions. Format of the packet used during isochronous transactions is also detailed.

Asynchronous Packets
The link layer controller (Link) is responsible for constructing the 1394 packets required to transmit data over the 1394 serial bus. Packet contents vary depending upon the transaction type (See Table 8-1 on page 167 for transaction type codes). Data comprising the packet is transferred to the physical layer controller (PHY) via an interface defined by the specification. The link to physical layer interface is defined in Chapter 11.
Figure 8-1 on page 166 illustrates the basic construct of primary request packets that are used during asynchronous packet transactions. Primary packets have a standard header format and an optional data block, whose presence depends on the amount of data to be transferred.

*Figure 8-1: Primary Asynchronous Packet Format*
Data Size

The data payload of asynchronous packets is limited to minimize the possible overrun of asynchronous transaction time into the isochronous transaction time. Recall that during a 125μs cycle a mix of isochronous and asynchronous transactions may be performed. The data payload limit corresponds to transmission speed as listed in Table 8-2 on page 168.
Write Packets

Three packets are defined for performing write transactions, including two forms of write request packets. Each of the following packet types is illustrated in the following pages and each field within the packet is defined:

- Write Quadlet Request packet (Figure 8-2 on page 169 & Table 8-3 on page 169).
- Write Data Block Request packet (Figure 8-3 on page 171 & Table 8-4 on page 172).
- Write Response packet (Figure 8-4 on page 173 & Table 8-5 on page 173).

<table>
<thead>
<tr>
<th>Cable Speed</th>
<th>Maximum Data Payload Size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100Mb/s</td>
<td>512</td>
</tr>
<tr>
<td>200Mb/s</td>
<td>1024</td>
</tr>
<tr>
<td>400Mb/s</td>
<td>2048</td>
</tr>
<tr>
<td>800Mb/s</td>
<td>4096</td>
</tr>
<tr>
<td>1.6Gb/s</td>
<td>8192</td>
</tr>
<tr>
<td>3.2Gb/s</td>
<td>16384</td>
</tr>
</tbody>
</table>
Chapter 8: Asynchronous

Figure 8-2: Write Request — Quadlet Format

<table>
<thead>
<tr>
<th>name</th>
<th>Abbrev.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Identifier</td>
<td>destination_ID</td>
<td>Combination of the bus address and physical ID of the node. Contains the address of the requesting node.</td>
</tr>
<tr>
<td>Transaction label</td>
<td>tl</td>
<td>A label specified by the requester that identifies this transaction. This value, if used, is returned in the response packet.</td>
</tr>
<tr>
<td>Retry code</td>
<td>rt</td>
<td>This code specifies whether this packet is an attempted retry and defines the retry protocol to be followed by the target node. 00 = Retry_1 (first attempt) 01 = Retry_X 10 = Retry_A 11 = Retry_B</td>
</tr>
</tbody>
</table>
Isochronous Packet

The Previous Chapter

Asynchronous transactions exist in three basic forms: reads, writes, and locks. The previous chapter discussed the asynchronous transaction types and related packets that are transmitted over the bus.

This Chapter

Isochronous transactions are scheduled so that they occur at 125µs intervals. This chapter discusses isochronous transaction issues and the format of the packet used during isochronous transactions.

The Next Chapter

Next, the various types of PHY packet are discussed. The role of each PHY packet is included, packet format is specified, and the fields within each packet are detailed.

Stream Data Packet

Isochronous transactions use a single data packet to perform a multicast or broadcast operation to one or more nodes. Target nodes are identified by a channel number rather than by a node ID and destination offset address. An isochronous transaction contains only a request phase with no acknowledgment and no response. An isochronous transaction uses a streaming data packet. The packet format is illustrated in Figure 9-1 on page 200 and each field is defined in Table 9-1 on page 201.
FireWire System Architecture

The stream data packet (know as the isochronous data block packet in the IEEE 1394-1995 specification) was supported solely for the isochronous bus period by the 1995 specification. The data stream packet has now been specified to occur during either the isochronous or asynchronous time by the 1394a supplement.

Prior to using an isochronous stream packet, the application must first obtain a channel number from the isochronous resource manager node. When a stream data packet is performed during the isochronous time, the application must all obtain the necessary bus bandwidth from the isochronous resource manager.

Figure 9-1: Format of an Isochronous Stream Packet

```
<table>
<thead>
<tr>
<th>data_length</th>
<th>tag</th>
<th>channel</th>
<th>tcode</th>
<th>sy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

header_CRC

data block

last quadlet of data block (padded if necessary)

data_CRC
```

msb (transmitted first)

lsb (transmitted last)
Chapter 9: Isochronous Packet

Table 9-1: Isochronous Stream Packet Components

<table>
<thead>
<tr>
<th>Component Name</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Length</td>
<td>data_length</td>
<td>Length can be any value from zero to all ones (FFFFh). When the data length is not a multiple of four bytes, then the talker must pad the last quadlet field with zeros.</td>
</tr>
<tr>
<td>Isoch Data Format Tag</td>
<td>tag</td>
<td>The value of 00b indicates that the isochronous data is unformatted. All other values are reserved.</td>
</tr>
<tr>
<td>Isoch Channel Number</td>
<td>Channel</td>
<td>Specifies the isochronous channel number assigned to this packet. Channel numbers are a simplified means of addressing. Channel numbers are assigned to a node for talking or listening.</td>
</tr>
<tr>
<td>Transaction code</td>
<td>tcode</td>
<td>The transaction code for an isochronous data block transaction is Ah.</td>
</tr>
<tr>
<td>Synchronization Code</td>
<td>sy</td>
<td>Application specific.</td>
</tr>
<tr>
<td>Header CRC</td>
<td>header_CRC</td>
<td>CRC value for header.</td>
</tr>
<tr>
<td>Data Block Payload</td>
<td>data_field</td>
<td>Data to be transferred by the talker. Last quadlet of data field must be padded with zeros if necessary.</td>
</tr>
<tr>
<td>Data Block CRC</td>
<td>data_CRC</td>
<td>CRC value for the data field.</td>
</tr>
</tbody>
</table>

Isochronous Data Packet Size

The maximum size of an isochronous transaction based on the 1394-1995 specification was limited to the maximum isochronous bus time, or 100µs. The 1394a supplement further constrains the maximum data block size to the values shown in Table 9-2 on page 202.

The specification describes a null isochronous stream packet. This packet has a data_length value of zero, making the size of this packet only 64-bit, or 2 quadlets. This is the only packet other than PHY packets with a length of 64-bits.
Isochronous Transaction Summary

The following illustrations summarize the standard and concatenated forms of isochronous transactions:

Figure 9-2: Non-Concatenated Isochronous Transactions

<table>
<thead>
<tr>
<th>Cable Speed</th>
<th>Maximum Data Payload Size (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100Mb/s</td>
<td>1024</td>
</tr>
<tr>
<td>200Mb/s</td>
<td>2048</td>
</tr>
<tr>
<td>400Mb/s</td>
<td>4096</td>
</tr>
<tr>
<td>800Mb/s</td>
<td>8192</td>
</tr>
<tr>
<td>1.6Gb/s</td>
<td>16384</td>
</tr>
<tr>
<td>3.2Gb/s</td>
<td>32768</td>
</tr>
</tbody>
</table>

Table 9-2: Maximum Data Payload for Isochronous Packets
Chapter 9: Isochronous Packet

Figure 9-3: Concatenated Isochronous Transactions - If Sent by Same Node
The Previous Chapter

Isochronous transactions are scheduled so that they occur at 125µs intervals. The previous chapter discussed isochronous transaction issues and the format of the packet used during isochronous transactions.

This Chapter

This chapter discusses the various types of PHY packets. The role of each PHY packet is discussed, packet format is specified, the fields within each packet are detailed.

The Next Chapter

The next chapter details the signaling interface between the link and PHY layer controller chips.

Overview

Transactions discussed to this point target memory-mapped address locations within the node or access a memory buffer identified by a channel number. These address locations reside physically in the link layer, transaction layer, or application layer. The PHY contains no memory-mapped address locations. Some packets however, are designed to access registers within the PHY. These register locations are not mapped within the 256TB of address space allocation to each node and can only be accessed by the local application or via a PHY packet. The PHY packet types defined by the IEEE 1394-1995 specification include:

- Self Identification (Self-ID) packet
- Link-On packet
- PHY Configuration packet
FireWire System Architecture

The 1394a supplement defines extended PHY packet types that include:

- Ping packet
- Remote Access packet
- Remote Reply packet
- Remote Command packet
- Remote Confirmation packet
- Resume packet

All PHY packets are transmitted at the base rate of 100Mb/s. This is to ensure that all PHYs are able to receive PHY packet regardless of the maximum transmission speed that they support.

PHY packets are used for various bus management functions the following sections discuss each packet type and describes the bus management functions controlled or affected by each packet.

**PHY Packet Format**

The general format of a PHY packet is illustrated in Figure 10-1. A PHY packet is eight bytes in length (one octet). The first four bytes (one quadlet) contain the PHY packet information. They are followed by four more bytes used for error checking. The second quadlet contains the 1’s complement (logical inverse) of the first quadlet.

*Figure 10-1: General Format of PHY Packet*

<table>
<thead>
<tr>
<th>Packet Identifier</th>
<th>phy_ID</th>
<th>Packet-specific information</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>logical inverse of first quadlet</td>
</tr>
</tbody>
</table>
Chapter 10: PHY Packet Format

A PHY recognizes a PHY packet if the total packet length is 64-bits. No other packet has a size of 64-bits except for the possibility of a steam packet that contains no data. A PHY recognizes the difference between a null stream packet and a PHY packet because the stream packet uses a 32-bit CRC, whereas, a PHY packet uses a 32-bit 1’s complement of the first quadlet.

The PHY packet types are differentiated by the first two bits of the packet as follows:

- 00 = PHY Configuration packet
- 01 = Link-on packet
- 10 = Self-ID packet

The extended packets are identified using the gap_count field of the PHY configuration packet format (“Extended PHY Packets” on page 214).

**Self-ID Packets**

During bus configuration each node must assign itself a node ID and notify other nodes of its serial bus capabilities. These actions take place during the self-identification process that occurs following a bus reset and tree identification. Each node broadcasts from one to three self-ID packets during the self-identification process. Note that this packet is created within the PHY layer.

**Self-ID Packet Zero**

Self-ID packet zero is illustrated in Figure 10-2. The format of self-ID packet zero is the same for 1394-1995 and the 1394a supplement. However power class fields within Self-ID packet zero are defined differently. Table 10-1 defines the fields for 1394-1995 and Table 10-2 on page 209 lists the new power class field definitions for 1394a. Note that self-ID packet zero permits identification of only three node ports (ports 0-2). If additional ports are implemented, then one or more self-ID packets is required.

The 1394a specification redefines the port status information within the self-ID packets. The four new states are defined as:

- 00b = port not present
- 01b = port not active (may be either suspended, disabled, or disconnected)
- 10b = port active and connected to parent node
- 11b = port active and connected to child node
### Table 10-1: Contents of the Self-ID Packet Zero — 1394-1995

<table>
<thead>
<tr>
<th>Field Code</th>
<th>Field Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Packet identifier</td>
<td>Transmitted at the beginning of the packet to identify this packet as a self-ID packet.</td>
</tr>
<tr>
<td>Phy_ID</td>
<td>Physical ID</td>
<td>Physical Identifier of the node sending this packet.</td>
</tr>
<tr>
<td>L</td>
<td>Link active</td>
<td>Set to indicate that Link and Transaction layers are active.</td>
</tr>
<tr>
<td>gap_cnt</td>
<td>Gap count</td>
<td>Current value of PHY_CONFIGURATION.gap_count field within the PHY register.</td>
</tr>
</tbody>
</table>
| sp         | PHY speed    | Nodes speed capabilities:  
00 = 98.304 Mb/s  
01 = 98.304 Mb/s and 196.608 Mb/s  
10 = 98.304 Mb/s, 196.608 Mb/s and 393.216Mb/s  
11 = Reserved |
| del        | PHY delay    | Specifies the maximum repeater delay across this node.  
00 = <= 144ns (~14/BASE_RATE)  
01 = Reserved  
10 = Reserved  
11 = Reserved  
Note: This field is reserved in the 1394a specification. PHY delay is obtained by reading the PHY delay and PHY jitter fields within the 1394a PHY registers.|
| c          | Contender    | When set and Link active is set, this node is a contender for the role of bus or isochronous resource manager. |
## Chapter 10: PHY Packet Format

**Table 10-1: Contents of the Self-ID Packet Zero — 1394-1995 (Continued)**

<table>
<thead>
<tr>
<th>Field Code</th>
<th>Field Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>pwr</td>
<td>Power class</td>
<td>Specifies power consumption and source characteristics:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000 = Node does not need bus power nor repeat bus power.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001 = Self powered &amp; provides 15W (minimum) to bus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010 = Self powered &amp; provides 30W (minimum to bus.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011 = Self powered &amp; provides 45W (minimum to bus</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100 = May be powered by bus and uses up to 1W.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101 = Powered by bus &amp; uses 1W. Additional 2W</td>
</tr>
<tr>
<td></td>
<td></td>
<td>110 = Powered by bus &amp; uses 1W. Additional 5W</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111 = Powered by bus &amp; uses 1W. Additional 9W needed</td>
</tr>
<tr>
<td>p0.. p2</td>
<td>Port number</td>
<td>Specifies port status:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00 = Port not present</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01 = No connection to other node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 = Connected to parent node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11 = Connected to child node</td>
</tr>
<tr>
<td>i</td>
<td>Initiated reset</td>
<td>This node initiated the current bus reset before receiving reset signaling. (Optional, if not used returns zero)</td>
</tr>
<tr>
<td>m</td>
<td>More packets</td>
<td>More packets follow this one to report additional port status.</td>
</tr>
</tbody>
</table>

**Table 10-2: Definition of Power Class Values Within “Pwr” Field of Self-ID Packet—1394a**

<table>
<thead>
<tr>
<th>POWER_CLASS Code (binary)</th>
<th>Power Consumption and Source Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Node does not require bus power nor repeat bus power.</td>
</tr>
<tr>
<td>001</td>
<td>Node is self-powered and provides 15W (minimum) to the bus.</td>
</tr>
<tr>
<td>010</td>
<td>Node is self-powered and provides 30W (minimum) to the bus.</td>
</tr>
</tbody>
</table>
The Previous Chapter

The previous chapter discussed the various types of PHY packet. The role of each PHY packet was discussed, the PHY packet formats was specified, and the fields within each packet were detailed.

This Chapter

This chapter details the signaling interface between the link and PHY layer controller chips.

The Next Chapter

The next chapter discusses the transaction retries that can occur when the recipient of a packet is busy (e.g., has a buffer full condition). Two retry mechanisms are defined by the 1394 specification: single and dual phase. Each type of mechanism is discussed.

Overview

The IEEE 1394-1995 specification describes the interface between the Link and PHY chips. This interface definition is included in the 1394-1995 Appendix and is labeled as informative (not a required implementation). However, the 1394a supplement requires that this interface be used if the node is implemented with separate PHY and Link layer components. Note that there is no requirement that a node be designed with separate Link and PHY chips. These functions could be integrated within the same silicon, in which case the interface between these functions is implementation specific. The motivation for requiring the standard interface is to promote interoperability between PHY and Link layer chips from different manufacturers.
The Interface Signals

Figure 11-1 on page 222 illustrates the synchronous interface between the link and PHY. The interface is used by both the link and PHY. The link layer chip initiates transactions by sending requests to the PHY to send a packet, and the PHY forwards packets that it receives from the 1394 cable to the link via the interface. The function of each Link/PHY interface signal is defined in Table 11-1 on page 223.

Figure 11-1: Interface Between the Link and PHY
### Chapter 11: Link to PHY Interface

**Table 11-1: Link/PHY Signal Interface**

<table>
<thead>
<tr>
<th>Name</th>
<th>Driven by</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D[0:7]</td>
<td>Link or PHY</td>
<td>Data — packet data is delivered via the data lines. The number of data lines used depends on the speed supported as follows: D[0:1] = 100 Mb/s, D[0:3] = 200 Mb/s, D[0:7] = 400 Mb/s.</td>
</tr>
<tr>
<td>Ctl[0:1]</td>
<td>Link or PHY</td>
<td>Control — defines the state of the interface when being driven by the Link or PHY (e.g. idle, sending status, transmitting or receiving a packet).</td>
</tr>
<tr>
<td>LReq</td>
<td>Link</td>
<td>Link Request — this serial interface is used by the link to initiate a request. Is also used to request access to local PHY registers.</td>
</tr>
<tr>
<td>SClk</td>
<td>PHY</td>
<td>49.152 MHz clock — the clock used to clock data between the PHY and Link.</td>
</tr>
<tr>
<td>LPS</td>
<td>Link</td>
<td>Link power status — indicates whether the link is powered or not.</td>
</tr>
<tr>
<td>Link On</td>
<td>PHY</td>
<td>Link has been powered-on via a link-on packet.</td>
</tr>
<tr>
<td>Direct</td>
<td>Neither</td>
<td>Direct connection between Link and PHY interface signals is implemented when this pin is asserted. When deasserted an isolation barrier is implemented.</td>
</tr>
<tr>
<td>Backplane</td>
<td>Neither</td>
<td>Pulled high for backplane PHY implementation.</td>
</tr>
<tr>
<td>Clk25</td>
<td>Neither</td>
<td>Pulled high if SClk is 24.576. (backplane environment only).</td>
</tr>
</tbody>
</table>
Sharing the Interface

The bi-directional control signals specify the type of transmission being made via the data lines. The type of transmission depends in part on whether the link or PHY is transmitting the data. The default owner of the PHY/link interface is the PHY. The link gains ownership of the interface via the LReq lines to transfer a packet to the PHY, which then transmits the packet over the cable. Ownership of the interface returns to the PHY after the link has completed sending the packet. The current state of the interface is defined by the control signals.

PHY Initiated Transfers

The PHY uses the interface to transfer information to the node's link layer controller. The information transferred includes:

- packets received from the serial bus that are forwarded to the link layer controller.
- contents of local PHY registers that have been requested by the link.
- status information which is reported to the link.

When the PHY owns the interface, one of four conditions may exist on the interface. The current state of the interface defines these conditions, which can be determined by monitoring the interface control lines (Ctl[0:1]). Table 11-2 lists the control signal definitions when the PHY owns the bus.

<table>
<thead>
<tr>
<th>Ctl[0:1]b</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>idle</td>
<td>No activity.</td>
</tr>
<tr>
<td>01</td>
<td>status</td>
<td>PHY is sending status information to link.</td>
</tr>
<tr>
<td>10</td>
<td>receive</td>
<td>Incoming packet is being transferred from PHY to link via data lines, or PHY generated data being sent to the cable and also to the link via the data lines (e.g. self-ID packets, remote command and reply packets).</td>
</tr>
<tr>
<td>11</td>
<td>grant</td>
<td>PHY is granting use of the bus to the link so that it can send a packet.</td>
</tr>
</tbody>
</table>
Chapter 11: Link to PHY Interface

Idle State

Interface is idle indicating that neither the link nor the PHY currently have data to transfer.

Status State

When the PHY signals status via the control lines, it is sending status information across the data lines. Status transmission occurs when one of several PHY or cable events have occurred. These events include:

- An arbitration reset gap has been detected
- A subaction gap has been detected
- A cable reset has been detected
- Cable power failure
- Looped topology has been detected during the Bus ID procedure
- Arbitration state machine has timed out
- Bias change at a disabled port has been detected

Also, PHY register data is returned to the link in response to a PHY register read request.

Receive State

When the receive state is being signaled, the PHY is transferring an incoming data packet from the 1394 bus to the link for decoding, or is forwarding PHY generated packets to both the cable and the link. PHY packets include self-ID packets, configuration packet, link-on packet, and all extended PHY packets.

Grant State

Grant is signaled to the link to notify it that it can begin transmitting a packet. This is done only after the link has issued a packet request via the LReq line and the PHY has obtained ownership of the 1394 bus.
The Previous Chapter

The previous chapter detailed the signaling interface between the link and PHY layer controller chips. This interface is now required by 1394a implementations of a PHY or link controller chip.

This Chapter

This chapter discusses transaction retries that occur when the recipient of a packet is busy (e.g. has a buffer full condition). Two retry mechanisms are defined by the 1394 specification: single and dual phase. Each type of mechanism is discussed. Software may also initiate retries for transactions that fail.

The Next Chapter

The next chapter overviews the configuration process comprising the initialization, tree ID, and self-ID phases. Once self-ID completes, additional configuration may optionally take place in the form of bus management activities that are also reviewed in this chapter.

Overview

Asynchronous transaction retry is supported by 1394 and may occur under three circumstances:

- Node is Busy
- Failed Packet Transfer
- Node is Locked

The specification defines a retry protocol to be used when the recipient of a packet is temporarily busy (e.g. due to a buffer full condition or a locked transaction is being performed). This form of retry is handled by hardware. When packet transmission fails due to an error condition, the requesting node may re-initiate the transaction under software control.
Busy Retry

Serial bus nodes employ separate request and response queues. When a node has a queue full condition (busy), the link layer must return an acknowledge packet that specifies the status of the packet just transferred. The acknowledge codes returned indicate if the node was busy, and if so which type of retry that the target node supports. Table 12-1 lists the retry codes that can be returned in the acknowledge packet. Note that retries may be performed for both request and response packet transfers.

Table 12-1: Retry Codes Returned by Busy Node

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>ack_busy_x</td>
<td>The packet was not accepted by the target node, but may be accepted when retried.</td>
</tr>
<tr>
<td>5</td>
<td>ack_busy_A</td>
<td>The packet was not accepted by the target node because the node was busy. The node will accept the data when not busy during the next retry phase A.</td>
</tr>
<tr>
<td>6</td>
<td>ack_busy_B</td>
<td>The packet was not accepted by the target node because the node was busy. The target node will accept the packet when not busy during the next occurrence of a retry B.</td>
</tr>
</tbody>
</table>

The First Packet Transmission Attempt

When the application wishes to initiate an asynchronous request or a response, it uses the “transaction data request” service or the “transaction data response” service, respectively. The transaction layer generates a “link data request” in response and specifies a retry code of retry_1 for the first attempt of sending a subaction. When the target node receives the packet, it may be able to accept the packet or it may initiate a retry in the event of a full queue. The retry behavior is described in the following sections.
Chapter 12: Transaction Retry

Single Phase Retry

A node that receives a packet for the first time will detect a retry code of retry_1. If the node is busy, it returns an acknowledge code of ack_busy_x indicating that it supports single phase retries. The transmitting node will send the packet again but this time it will return the retry_x code that it received in the acknowledgment packet. Each time the transaction is retried a retry code of retry_x is used. The transaction is retried until it is successful or until the retry limit is exceeded. The retry limit is specified in the BUSY_TIMEOUT register (offset 210h in CSR space) illustrated in Figure 12-1. Note that the maximum number of single phase retries is limited to sixteen by the 4-bit retry_limit field.

Single phase retry provides no scheduling mechanism to handle older transactions that may have been retried many times prior to handling newer ones.

Sending-Node Retry Behavior (Outbound Retry)

The state machine diagram in Figure 12-2 illustrates the actions taken by the transaction layer of a node that is sending packets to a sometimes-busy node. Two states are defined in the state diagram Outbound Single Retry zero (OSR0): “Ready to Send” and Outbound Single Retry one (OSR1): “Pending Retry.” The following list describes each of the state transitions.

- **OSR0** — The initial entry into the Ready to Send state occurs when the link issues an initialization or reset control request to the transaction layer.
- **OSR0 to OSR0** — This transition occurs when the subaction that has been just sent receives a normal acknowledgment (i.e. an acknowledgment other than any type of ack_busy).
- **OSR0 to OSR1** — The subaction just sent receives an ack_busy acknowledgment of any type.
- **OSR1 to OSR1** — The subaction that was just retried receives one of the three forms of ack_busy. The retry count has not been exceeded. And the transaction layer has not chosen to requeue the pending retry.
FireWire System Architecture

- OSR1 to OSR0 — This transition may be taken as a result of three separate conditions:
  1. The transaction layer receives an acknowledge code other than ack_busy. And the packet has been sent successfully.
  2. The retry count has been exceeded and the transaction layer terminates further retries of this packet. The failed transaction is reported to the application.
  3. The transaction layer receives one of the forms of ack_busy in response to the packet just retried. The retry count has not yet expired. And the transaction layer chooses to requeue to pending retry.

Receiving-Node Retry Behavior (Inbound Retry)

The state machine diagram in Figure 12-3 illustrates the actions taken by the transaction layer of a sometimes-busy node when it receives packets from other nodes. Two states are defined by the state diagram: Inbound Single Phase Retry zero (ISR0): Accept All and Inbound Single Phase Retry (ISR1): Busy All. The state transitions are described below.

- ISR0: This state is entered when the link layer sends an initialization or reset control request to the transaction layer. In this state the node is ready to
accept primary packets from other nodes.

- **ISR0 to ISR0** — This transition occurs when the node receives a primary packet and the transaction layer resources are available to accept this packet. The transaction layer returns a data response to the link and specifies the appropriate acknowledge code (not busy).
- **ISR0 to ISR1** — The transaction layer resources are no longer available (i.e. the transaction layer is now busy).
- **ISR1 to ISR1** — The transaction layer receives a primary packet and the resources are not available; thus, a link data response is issued to return an acknowledge packet set to ack_busy_X.
- **ISR1 to ISR0** — The transaction resources have become available (i.e. the transaction layer is no longer busy).

*Figure 12-3: Inbound Retry State Machine — Single Phase*

---

**Dual Phase Retry**

Just as with single phase retry, the initial access sends a retry code of retry_1. However, nodes that support dual phase retries return a retry acknowledge code of either ack_busy_A or ack_busy_B. When the initiator of the packet attempts packet transmission again, it specifies the same retry code as it received in the acknowledge packet.

To explain the actions of the target node, assume that it just received a packet that must be retried. The busy node returns an ack_busy_A code and waits for the retry to occur. However, if other packets target this same node with retry codes of retry_1, the target returns ack_busy_B retry codes. Since the A packet
The Previous Chapter

The previous chapter discussed transaction retries that occur when the recipient of a packet is busy (e.g. has a buffer full condition). Two retry mechanisms are defined by the 1394 specification: single and dual phase. Each type of mechanism is discussed.

This Chapter

This chapter overviews the configuration process comprising the initialization, tree ID, and self-ID phases. Once self-ID completes additional configuration may optionally take place in the form of bus management activities that are also reviewed in this chapter.

The Next Chapter

The next chapter details the bus reset phase of the cable configuration process.

Overview

1394 device configuration occurs locally on the serial bus without the intervention of a host processor. Each time a new device, or node, is attached or removed from the serial bus, the entire bus is reset and reconfigured. This chapter overviews the cable configuration process and subsequent chapters detail each step in the procedure.

Three primary procedures must be performed during cable configuration:

- Bus initialization
- Tree identification
- Self identification
FireWire System Architecture

Since cable configuration does not require interaction with the host processor, no single node can be identified as a root node based simply on the serial bus physical topology. Rather, a single node on the serial bus must be identified, via the Tree ID process, as the root node. The root node performs certain bus management functions for all devices residing on the bus. For example, the root node must take responsibility for establishing the intervals at which isochronous transactions are to be performed across the bus.

Since it is unknown at configuration time whether a given node supports 100, 200, or 400Mb/s transfers, all configuration transfers take place at the base rate of 100Mb/s.

During cable configuration the bus is reset and all 1394 bus traffic stops while the new topology is determined (during tree-ID) and while all nodes assign themselves a node ID (during Self-ID). All asynchronous transactions that are pending completion when cable configuration begins must be discarded and requeued by the local application once configuration completes. This is necessary because the node ID values used to target a given node may change due to a topology change. Consequently, all asynchronous transactions pending prior to reset may target the wrong node following a reset. Since isochronous transfers identify target(s) based on channel numbers (which are not affected by cable configuration), they can resume where they left off after configuration completes.

The speed at which the cable configuration process completes is obviously important since all pending transactions stopped during configuration. Figure 13-1 illustrates reset, tree-ID, and self-ID timing for the 1394-1995 environment. The following chapters detail each phase of the cable configuration process.

Figure 13-1: Overall Cable Configuration Time

![Figure 13-1: Overall Cable Configuration Time](image-url)
Chapter 13: Configuration Process

Bus Initialization (Bus Reset)

Reset occurs when power is applied to (e.g. initial power up) or removed from a node or when a node is attached to or removed from the 1394 bus. This forces all nodes to return to their initialized states. If the bus has been configured previously, then the topology will have been established. However, reset also clears all topology information from the nodes. Figure 13-2 illustrates a family of 1394 nodes with topology established and a subsequent view of the topology immediately following reset. Note that after reset all topology information is cleared, leaving each node at the same hierarchy in the topology. Reset initializes the bus and prepares each node to begin the tree identification process.

Figure 13-2 also illustrates that some nodes connect to only one other node, while others connect to one or more nodes. Nodes having a single port are termed leaf nodes (nodes A and E), and nodes with two or more ports are termed branches (nodes B, C, and D). The node labels A-E are provided for reference purposes only. Each node assigns a number to each of its ports, thereby providing a unique label for port identification. Port numbers are used during the self_ID process to determine the order (lowest to highest number) that ports are identified and also during normal arbitration as a tie breaker.

Reset signaling is sensed by the PHY layer of each node via the arbitration signal lines. Each node receiving RESET propagates reset signaling to the other nodes that it attaches to (if any), thereby ensuring that all nodes are reset. After each node is reset it enters the idle state and waits for a sufficient period of time to ensure all nodes have received a RESET and have entered their idle states. From the idle state all nodes begin the tree identification process.
Tree Identification (The Family Tree)

The tree identification process defines the topology of the bus based on the new family of devices that now reside there. After tree identification, one node will have gained status as the root node. Prior to tree identification, each node knows whether it connects to a single 1394 node (when it is a single leaf on the bus) or more than one node (when it is a branch on the bus). Any node, a leaf or a branch, may become the root node. Figure 13-3 illustrates a series of nodes (branches and leaves) connected to the bus.
Chapter 13: Configuration Process

The tree identification process results in each port being identified as either a parent or a child. A node having a parent port designation means that the node at the other end of the cable is closer to the root, and that node will have identified its port as a child. A port identified as a child node points to a child node that is further away from the root. Any node having all of its ports identified as child ports becomes the root hub. A node can become the root node regardless of where it connects into the network of nodes.

Figure 13-4 illustrates the same family of nodes pictured in Figure 13-3, but after tree identification has completed. In this example, it is assumed that node D has been identified as the root node, thus both of its ports are identified as child ports. All other nodes have at least one of their ports identified as a parent port, meaning that there is another node higher in the topology hierarchy. Once tree identification has completed, the root node initiates the self-identification process.

Chapter 15 details the tree identification process and defines how the root node is selected.
The Previous Chapter

The previous chapter provided an overview of the configuration process. The process comprises Initialization, Tree ID, Self-ID phases, and bus management activities.

This Chapter

This chapter details the bus reset phase of the cable configuration process. Initialization begins with the assertion of a bus reset by a given node on the bus. This chapter discusses the reset enhancements introduced by the 1394a supplement: debouncing the bias change detection, arbitration (short) bus reset, and new timing parameters.

The Next Chapter

Following bus initialization, the tree ID process begins to determine which node will become the root. The next chapter details the protocol used in determining the topology of the nodes.

Overview

Bus reset forces all nodes into their initialization state, thereby initiating the configuration process. Bus reset is initiated under software control or as a result of hardware events as discussed below. Note that assertion of reset signaling by a node does not terminate a transaction currently being performed. When the transaction ends, all other nodes will have their drivers disabled, thus the reset signaling will be detected by all nodes.
FireWire System Architecture

When each node receives reset it clears all information related to the bus topology. However, the PHY of each port latches and saves connection status associated with each of its ports (i.e. whether a device is currently attached to each port or not). If a device is removed or attached to a port during reset, the change will be detected at the end of the initialization phase, thereby causing the node to signal reset again, forcing all nodes back into reset.

Sources of Bus Reset

Bus reset is initiated under the following circumstances:

- Bus reset is signaled when power status changes at the PHY.
- Bus reset signaled by an attached node.
- Node attachment or removal.
- PHY detects MAX_ARB_STATE time-out. That is, a PHY stays in any state (except Idle, Tree ID Start, or other state that has an explicit time-out defined) for longer than the MAX_ARB_STATE_TIME.
- PHY receives a bus reset request initiated by software.

The following sections describe each type of reset.

Power Status Change

When a locally powered PHY receives a power reset or detects a change in its power state, it must signal a bus reset. No bus reset is required if only the link layer power state changes.

Bus Reset Signaled by Attached Node

When a port detects reset signaled by an attached node it must repeat reset signaling to its other connected ports. This node may be in the process of repeating a packet when the reset is detected at one of its ports. Even if a packet is currently being repeated to that port, the arbitration comparators will detect reset signaling due to 1’s dominance decoding of the arbitration comparator outputs.
Chapter 14: Bus Reset (Initialization)

Node Attachment or Removal

If a node’s PHY recognizes that another node has been attached to or removed from one of its ports, it must signal reset to all of its active ports. The PHY must also signal the bus reset event to the link. Node attachment or detachment is detected by a port receiver when it detects a change in the bias voltage. A variety of conditions exist that determines the exact behavior of a node, after it has detected a bias change. These conditions and behaviors are discussed in “Effects of Bus Reset” on page 277.

MAX_ARB_STATE_TIME Expires

The MAX_ARB_STATE_TIME parameter applies to all states within the PHY with the exception of those listed below. When the PHY remains in a given state long enough for the MAX_ARB_STATE_TIME to be detected (200µs min. to 400µs max.), it must signal reset. The PHY states excluded from the MAX_ARB_STATE_TIME limit include:

- Bus Idle
- Tree ID Start
- Any state that has an explicit time-out value defined

Software Initiated Bus Reset

An application can initiate a bus reset by making a control_request to the node controller software, which in turn issues the request to the link and PHY. In response, the PHY sets its IBR (initiate bus reset) bit, causing the reset to be signaled. When reset signaling ends a reset_complete confirmation is returned to the link and forwarded on to the local application via the transaction layer application.

Bus Reset Signaling

When the PHY receives a reset request, it drives 1s on TPA and TPB for each of its ports. All other nodes receiving reset propagate, or repeat, reset signaling to their other ports. This action ensures that all nodes receive RESET. Figure 14-1 on page 276 illustrates reset being signaled and detected by two attached nodes. Note that even though a child node is currently signaling a TX_REQUEST in an
FireWire System Architecture

attempt to gain bus ownership, the reset (1,1) prevails due to 1’s dominate decoding of the received signals.

The duration of reset must be sufficiently long to permit a transaction being performed to complete. The longest reset timing requirement is a minimum reset duration of 167µs.

*Figure 14-1: Reset Signaling and Detection*
Chapter 14: Bus Reset (Initialization)

Effects of Bus Reset

Bus reset results in a variety of actions being taken within nodes. The primary effects include:

- Topology information is cleared at each port
- Some PHY register values return to defaults
- CSR register values affected

The effects of bus reset also depend on the source of the reset. For example, when power is cycled, a bus reset is performed and all register values within the PHY and link are cleared or returned to their initial values. Bus reset for any other reason has a less drastic effect, resulting in some register fields being preserved. The following discussion describes the effects of a bus reset not resulting from power being cycled.

Topology Information Cleared

Figure 14-2 illustrates how each node would view itself following RESET. Note that all nodes have the same peer relationship to each other. Following the completion of bus reset, all nodes enter their tree ID state and the topology is re-established.

Figure 14-2: Nodes After Reset Have No Sense of Bus Topology

PHY Register Changes

Some fields within the port and PHY registers either lead to the generation of bus reset or are directly affected by a bus reset while others remain unaffected. Following is a description of the fields related to bus reset.
15 Tree Identification

The Previous Chapter

The previous chapter detailed the initialization phase of the configuration process. Initialization begins with the assertion of a bus reset by a given node on the bus.

This Chapter

Following bus initialization, the tree ID process begins to determine which node will become the root. This chapter details the protocol used in determining the topology of the serial bus.

The Next Chapter

The next chapter focuses on the self-ID process. During self-ID all nodes are assigned addresses and specify their capabilities by broadcasting self-ID packets.

Overview

Following bus initialization, nodes begin the tree identify phase to identify the root node and the topology of all attached nodes. The tree ID process results in all ports being designated as either child or parent ports. A child port connects to a node further away from the root, while a parent port connects to a node closer to the root.
Tree ID Signaling

Before discussing the tree ID process a review of the tree ID signaling may be helpful. All connected nodes perform Tree ID signaling via the strobe and data drivers and their respective arbitration comparators as illustrated in Figure 15-1. The strobe is delivered via TPA and is received by the attached node on TPB, while the state driven on the data is delivered via TPB and is received on TPA.

All nodes use the arbitration mechanism to communicate with other nodes that are attached directly to their ports. The tree ID process uses two of the arbitration line states defined as:

- Parent_Notify
- Child_Notify

These signal states are used to determine if a given node is closer to or further from the root node. The line states that are driven are listed in Table 15-1.

A Parent_Notify is signalled by driving a “0” onto the TPA, while leaving TPB in the idle or undriven state (Z). Child_Notify is signaled by driving a “1” on TPA and leaving TPB in the “Z” state. As illustrated in Figure 15-1 on page 287, the signals driven onto TPA are received and detected on TPB of the attached node’s arbitration comparators, and signals driven on TPB are received and detected on TPA by the arbitration comparators. The following discussion details the Tree ID process.

<table>
<thead>
<tr>
<th>Parent_Notify</th>
<th>Child_Notify</th>
</tr>
</thead>
<tbody>
<tr>
<td>TPA 0 (Strobe)</td>
<td>TPA 1 (Strobe)</td>
</tr>
<tr>
<td>TPB Z (Data)</td>
<td>TPB Z (Data)</td>
</tr>
</tbody>
</table>

The Tree ID Process

The Tree ID process begins with one or more nodes signaling Parent_Notify to their probable parents. Only those nodes that have received a Parent_Notify on all but one of their ports can signal a Parent_Notify. Immediately following reset this is true of only leaf nodes, since they have a single port. Thus, all leaf nodes immediately signal Parent_Notify to the attached nodes. Branch nodes that
receive a Parent_NOTIFY on one of their ports mark that port as a “child” port to signify that the attached node is further away from the root node. In response to the Parent_NOTIFY, branch nodes also return a Child_NOTIFY to the attached leaf node. However, a branch node will not signal the Child_NOTIFY until all but one of its ports have received Parent_NOTIFY. A leaf node that receives a Child_NOTIFY marks its port as a parent port which signifies that the attached node is closer to the root node. Once all nodes have identified all other attached nodes as either children or parents, the Tree ID process is complete.

Figure 15-1: Strobe and Data Lines Used During Tree ID

Leaf Nodes Try to Find Their Parents

The following discussion details the handshake performed between a pair of leaf and branch nodes during the Tree_ID process. Note that all leaf and branch nodes that connect to each other perform the same handshake. Immediately following Bus Initialization, all leaf nodes signal Parent_NOTIFY via TPA=0 and
FireWire System Architecture

TPB=Z as pictured in Figure 15-2 on page 288. At this time, the branch node is not signaling any line state, thus its data and strobe drivers are in a high impedance (Z) state and the line states driven by the leaf node are unaffected by the branch node. Since the twisted pair signal lines are cross-wired between attached nodes, the branch node observes the Parent_Notify at its arbitration comparators as TPA=Z and TPB=0. At the same time, the leaf node’s arbitration comparators observe the line states that its own drivers are signaling (TPA=0 and TPB=Z).

Parents Identify Their Children

The branch node, having received a Parent_Notify on one of its ports, recognizes that a leaf is attached to this port and marks the port as a child port. In
response, the branch node signals a Child_Notify via TPA=1 and TPB=Z when it recognizes that all of its ports or all but one of its ports have received Parent_Notify. While the Child_Notify is being signaled by the branch node, the leaf continues to signal Parent_Notify. Figure 15-3 on page 289 illustrates the states driven by both nodes and the resulting line states. The resulting line states observed at the leaf’s arbitration comparators are TPA=0 and TPB=1. The branch node also detects the Child_Notify as TPA=1 and TPB=0.

Upon detecting the Child_Notify, the leaf node marks its port as a parent port (i.e. it attaches to a node that is closer to the root). Having been identified as a child port, the node withdraws its Parent_Notify, which is viewed by the branch node as confirmation that the Child_Notify was received and accepted by the leaf node (i.e. the branch node observes the line state change from TPA=0 and TPB=0 to TPA=1 and TPB=Z). The leaf node’s role in the Tree ID process is now finished and the first phase of the Tree ID process is complete.

Figure 15-3: Branch Node Signaling Child_Notify
16 Self Identification

The Previous Chapter

Following bus initialization, the tree ID process begins to determine which node will become the root. The previous chapter detailed the protocol used in determining the topology of the serial bus.

This Chapter

This chapter focuses on the self-ID process. During self-ID all nodes are assigned addresses and specify their capabilities by broadcasting self-ID packets.

The Next Chapter

The next chapter describes the role of the cycle master node and defines how the cycle master is identified and enabled.

Overview

During the self-identification process configuration of the nodes begins. The following actions are performed during self-ID:

- Physical IDs are assigned to each node.
- Neighboring nodes exchange transmission speed capabilities.
- The topology defined during tree identification is broadcast.

During self-ID the root hub issues one self-ID grant signal for each node within the network. As each node receives its arbitration grant, it performs self-identification by assigning itself a physical ID and returning one or more self-identification packets. This process uses arbitrary port numbers that were assigned to each node during design time. These numbers are used to specify the order in which nodes attached to these ports will be assigned their physical IDs.
FireWire System Architecture

All signaling and packet transmission is done at the base rate (100Mb/s) since the speed capabilities of each node is unknown until the self-ID process has completed.

Self-Identification Signaling

Arbitration signaling states are used during the self-ID process. The signal states transmitted as listed in Table 16-1.

Table 16-1: Line States that Signal Parent- and Child-Notify

<table>
<thead>
<tr>
<th>TPA (Strobe)</th>
<th>DATA_PREFIX</th>
<th>IDENTIFICATION_DONE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>TPB (Data)</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Self-ID begins with the root signaling arbitration grant to its lowest numbered port and data prefix to its other ports. The following sections describe the entire self-ID process.

Physical ID Selection

At the start of the self-ID process, each node has identified whether each port points to a child or parent port as illustrated in Figure 16-1 on page 307. At this time no knowledge exists within any given node regarding the capabilities of its neighboring nodes or regarding the topology. The root node must determine how many nodes exist in the network and ensure that each is assigned a unique physical identifier. The following sections detail the process of assigning a physical ID to each node.

First Physical ID is Assigned

The self-ID process begins with the root port issuing an arbitration grant (TPA=Z and TPB=0) to its lowest numbered port and data prefix to its other ports. Arbitration grant is signaled by each branch node to its lowest numbered port until a leaf node is reached. Figure 16-2 illustrates the propagation of arbitration grant. In this example, when leaf node A receives the arbitration grant, it assigns itself a physical ID of zero (Phy 0). The physical ID assigned comes from the current self-ID count.
Chapter 16: Self Identification

Self-ID Count

Each node tracks the number of self-ID packets that are broadcast during the self-ID process. The self-ID count within all nodes is initialized to zero after reset. Nodes increment their self-ID count after each self-ID packet is broadcast.
Branch Nodes Signal Arbitration Grant & Data Prefix

Refer to Figure 16-2 on page 308. Note that as each branch node receives the arbitration grant, it checks its ports to determine which have been identified, if any. Since the self-ID process has just begun none of the ports have been identified yet. Each branch node (nodes C and B in this example) signals arbitration grant to its lowest numbered unidentified port. Each branch node also signals data prefix to its other ports, including back upstream to the node signaling the arbitration grant.
Chapter 16: Self Identification

When branch nodes signal data prefix upstream toward the root node, the upstream node continues to signal arbitration grant downstream. Consider the action taken by branch Node C when it receives arbitration grant from the root node. Figure 16-3 on page 309 illustrates the arbitration grant line state being signaled by the root node (TPA=Z and TPB=0) and the data prefix being signaled by node C (TPA=0 and TPB=1). The root node’s arbitration comparators will detect the arbitration grant that it is signaling; thus, when the data prefix is driven at the opposite end of the cable, the root node detects a change in the line state. The resulting line state (TPA=1 and TPB=0) observed by the root, is interpreted as receipt of a data prefix (Rx_DATA_PREFIX). When the root node detects the data prefix it removes arbitration grant signaling and leaves only the data prefix being driven on the cable.

Figure 16-3: Identified Node Starts Self-ID Packet Transmission with Data Prefix
The Previous Chapter
The previous chapter focused on the self-ID process. During self-ID all nodes are assigned addresses and specify their capabilities by broadcasting self-ID packets.

This Chapter
This chapter describes the role of the cycle master node, and defines how the cycle master is identified and enabled.

The Next Chapter
Next, the isochronous resource manager is discussed: how it is identified and enabled, and the nature of its role in the serial bus environment.

Overview
Isochronous transfers are guaranteed a constant bus bandwidth based on allocation of the number of bytes to be transferred during 125µs intervals. The root node is responsible for specifying the 125µs interval and marks the beginning of the next series of isochronous transactions by broadcasting a cycle start packet as illustrated in Figure 17-1.

Determining and Enabling the Cycle Master
A node that is cycle master capable must:

- be isochronous capable
- implement the BUS_TIME register
- be able to generate cycle start events based on an 8KHz clock that is synchronized to the CYCLE_TIME register and broadcast cycle start packets.
- set the cmc bit in the BUS_INFO_BLOCK
The cycle master must be the root node. Following the self-ID process, the root node and the isochronous resource manager will be known. If a bus manager is present it will verify that the root node is cycle master capable and, if so, enable it; otherwise, the isochronous resource manager will perform this function. To determine if the root is cycle master capable, the cmc bit within the root node’s BUS_INFO_BLOCK register will be set. If so, the root node is enabled to perform the cycle start by setting the cmstr bit in the STATE_SET register.

If the root node is not cycle master capable, other nodes are checked for cycle master capability. When a capable node is found it is selected to become the new root node. This is accomplished by broadcasting a PHY configuration packet with the force root bit “R” set to 1 and the root ID value set to the node ID of the target node. The selected node will set its root holdoff bit (RHB), while all other nodes (those not selected by the root ID value) will clear their RHB bit.

**Cycle Start Packet**

The cycle start packet format is illustrated in Figure 17-2 on page 331. For a description of the fields within the cycle start packet see Table 8-16 on page 196. This packet is broadcast by the cycle master at the beginning of each new cycle (nominally 125µs). The start of each cycle is synchronized to the cycle master’s CYCLE_TIME register. The cycle master delivers the contents of its cycle time register in the cycle start packet. As a result, if the cycle start packet is delayed due to the previous cycle stretching beyond the nominal 125µs cycle time, the timing variation will be visible to all isochronous nodes as illustrated in Figure 17-3 on page 331.
Chapter 17: Cycle Master

Figure 17-2: Cycle Start Packet Contains Value of Cycle Master's CYCLE_TIME Register

<table>
<thead>
<tr>
<th>msb (transmitted first)</th>
<th>tl</th>
<th>rt</th>
<th>tcode</th>
<th>pri</th>
</tr>
</thead>
<tbody>
<tr>
<td>destination_ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>source_ID</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination_offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>destination_offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>cycle_time_data (from CYCLE_TIME register)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>header_CRC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>lsb (transmitted last)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 17-3: Cycle Time Variation Included in Cycle Start Packet

Cycle n-1

nominal cycle period = 125µs

Packet A

Cycle Start

data=x

Packet B

Cycle Sync

data=y

Packet C

Cycle Start

delay=x

Cycle n

Cycle n+1

delay=y

= Isochronous Transactions

= Asynchronous Transactions
The Previous Chapter

The previous chapter described the role of the cycle master node, and defined how the cycle master is identified and enabled.

This Chapter

This chapter describes the role of the isochronous resource manager: how it is identified and enabled, and how other nodes interact with it.

The Next Chapter

Next, the bus manager function is described including topology map generation and access, speed map generation and access, and power management.

Overview

Following a bus reset, all traffic on the bus is terminated and all nodes perform the initialization sequence consisting of reset, tree-ID, and self-ID procedures. If this is the initial Reset due to power on, each node wishing to perform isochronous transfers must obtain an isochronous channel number and request the amount of bus bandwidth that it requires. The isochronous resource manager fulfills the role of keeping track of channel numbers and bus bandwidth that have been allocated.

If the reset occurs after isochronous traffic has started (e.g. due to attachment of new node), all bus traffic resumes as quickly as possible. Asynchronous transactions can resume immediately upon completion of the self-ID process, as well as isochronous transactions in most instances. However, isochronous transactions are delayed if the root node changes. In this case, isochronous transactions cannot start again until isochronous resources are verified and a cycle master is selected to re-initiate isochronous traffic.
Determining the Isochronous Resource Manager

Any node residing on the bus may have the ability to perform the role of isochronous resource manager (IRM). Nodes capable of becoming the isochronous resource manager must indicate their ability to fulfill this role by setting the “l” (link active) and “c” (contender) bits in packet zero of their self-ID register. This makes a given node a contender for the role of IRM. See Figure 18-1.

*Figure 18-1: Contender Nodes Must Set Bits l and c in Their Self-ID Packets*
Chapter 18: Isochronous Resource Manager

All nodes contending for the role of IRM must monitor all self-ID packets to determine if another node is also vying for the position of IRM. The competition is won by the contender having the highest value physical ID. During the self-ID process, a physical ID of zero is assigned first followed by consecutively higher IDs. Thus, a node contending for the role of IRM recognizes that it is out of the running if, after sending its own self-ID packet, it recognizes that a later self-ID packet specifies another node as a contender. Note that the highest value physical ID always belongs to the root node and its self-ID packet is always sent last. Thus, in many instances, the root node will become the IRM (i.e. when it is IRM capable).

Minimum Requirements of Isochronous Resource Managers

To be a contender for the role of IRM, a node must fulfill a set of minimum requirements:

- Support isochronous transactions as either talker or listener.
- Link and Transaction layers must be active during configuration process.
- Implement General ROM to support Bus_Info_Block with IRMC bit set.
- Implement the Bus Manager ID register.
- Implement bus bandwidth allocation register.
- Implement channel allocation register.

The isochronous resource manager provides allocation registers whose locations are known to all nodes needing to perform isochronous transactions. Nodes wishing to perform isochronous transfers must access these registers to acquire a channel number and bus bandwidth prior to performing any isochronous transfers.

Enabling the Cycle Master

The isochronous resource manager (IRM) may also be required to enable the cycle master so that it can begin transmitting cycle start packets. Following a power reset, the “cmstr” bit within the bus_depend field of the STATE register (See Figure 21-3 on page 367) will be cleared, which disables cycle master functionality. The IRM, in the absence of the bus manager, is responsible for enabling the root to perform the cycle master functions. The IRM determines that the bus manager is absent if the value of the BUS_MANAGER_ID register remains at 3Fh for greater than 625ms after bus reset.
FireWire System Architecture

In the event that a bus or command reset occurs, the cycle master should keep the “cmstr” bit set so that it can automatically resume broadcast of cycle start packets. However, if the cycle master recognizes that following a bus reset and tree ID process that it is no longer the root, it must clear the “cmstr” bit. Consequently, the new root node must be enabled as the cycle master so it can start generating cycle start packets.

Resource Allocation Registers

Figure 18-2 on page 336 shows the location of the CHANNELS_AVAILABLE and BANDWIDTH_AVAILABLE register within the node space of the Isochronous Resource Manager. The BANDWIDTH_AVAILABLE register is mapped at offset 224h from the beginning of the Serial-Bus dependent address space, and the CHANNELS_AVAILABLE register is mapped beginning at offset 220h. When a node accesses these registers to obtain isochronous resources, it must perform the access using lock transactions.

Figure 18-2: Location of CHANNEL_ALLOCATION & BUS_BANDWIDTH Registers.
Chapter 18: Isochronous Resource Manager

Channel Allocation

Nodes wishing to perform isochronous transfers must first obtain an isochronous channel number via the CHANNELS_AVAILABLE register.

Channels Available Register Format

This 64-bit register provides a bit map where each bit corresponds to one of the 64 possible isochronous channels supported by the serial bus as illustrated in Figure 18-3 on page 337. All bits are initialized to a value of one, thus indicating that none of the channels has been allocated.

Accessing the Channels Available Register

A node wishing to obtain an isochronous channel must first read the current register value to determine the next consecutive channel available. Note that the CHANNELS_AVAILABLE register is initialized to all ones, indicating that all channels are available. Next, the lock (compare and swap) transaction is used to request the next available channel. The lock transaction is used since more than one node may simultaneously attempt to request a channel. If no other node claims a channel number between the initial register read and the subsequent lock operation, then the lock transfer for this node will be successful; otherwise, the lock transfer will not succeed.

Figure 18-3: Format of the CHANNELS_AVAILABLE Register

<table>
<thead>
<tr>
<th>msb</th>
<th>channels_available_hi format</th>
<th>lsb</th>
</tr>
</thead>
<tbody>
<tr>
<td>msb</td>
<td>channels_available_lo format</td>
<td>lsb</td>
</tr>
</tbody>
</table>

ch 63  ---------------  channel bit assignment  ---------------  ch 32
ch 31  ---------------  channel bit assignment  ---------------  ch 0
19  Bus Manager

The Previous Chapter
The previous chapter described the role of the isochronous resource manager: how it is identified and enabled, and how other nodes interact with it.

This Chapter
In this chapter, the bus manager function is described including topology map generation and access, speed map generation and access, and power management.

The Next Chapter
The next chapter discusses the bus management services that are used by the bus manager and isochronous resource manager to perform their bus management roles.

Overview
One node residing on the serial bus may be selected to provide serial bus services for the benefit of the community of all nodes residing on the bus. Whether bus management is performed and the extent to which it is performed varies depending on the capability of nodes residing on the bus. Several possibilities exist:

- Bus is fully managed — at least one node on the bus is bus manager capable, thereby providing complete bus management facilities.
- Bus is partially managed — no bus management capable node is present on the bus, but at least one node is isochronous resource manager capable. Partial bus management capability is performed by isochronous resources management capable nodes.
- Bus is unmanaged — none of the nodes residing on the bus is bus manager capable or isochronous resource manager capable.
FireWire System Architecture

The bus management services that may be provided include:

- publishing a topology map that can be accessed by other nodes.
- publishing a speed map that can be read to find the maximum speed for each cable segment that is attached between two nodes.
- enabling the cycle master.
- power management control.
- optimizing bus traffic.

The node that performs bus management duties may reside anywhere on the serial bus. It collects information during the self-identification sequence as each node broadcasts its self-ID packet. The bus manager node uses this information to build topology and speed maps that other nodes can access. More than one node may be a candidate for the role of bus manager and these nodes must also monitor self-ID packets in the event they are selected to handle the role of bus manager. Furthermore, any node wishing to access the topology or speed maps must also monitor the self-ID packets so it can determine which node will perform the role of bus manager, and knowing its physical ID it will be able to access the topology and speed maps.

Determining the Bus Manager

At the conclusion of the Self-ID stage of bus configuration, the isochronous resource manager will have been identified. The last node to send its self-ID packets with the “L” and “C” bits set (the last contender) wins the role of isochronous resource manager. An isochronous resource manager may optionally be bus manager capable. A bus manager capable node differentiates itself from isochronous resource manager only nodes by performing a locked compare and swap transaction to the BUS_MANAGER_ID register within the IRM. The first node that successfully updates the BUS_MANAGER_ID register with its own physical ID wins the role of bus manager.

Other nodes read the BUS_MANAGER_ID register to obtain the node ID of the bus manager. If the value in the register is 3Fh, then no bus manager has claimed the role.
Power Management

Some nodes may require bus power to operate. If bus power is available, it is supplied by one or more nodes. When nodes are attached they must have their PHY powered in order for the node to function. Other node components may also require bus power such as the link layer, node controller, and unit related hardware. Any node hardware other than the PHY and node controller must remain powered off until configuration completes. The PHY and node controller functionality must be powered so that the PHY can power up the rest of the node under direction from the bus manager or isochronous resource manager (in the absence of a bus manager node). The following discussion is based on the 1394-1995 specification. The 1394a supplement extends the definition of power management and is discussed in the next chapter.

Power Management by Bus Manager Node

The bus manager obtains power class information from each node when the PHY sends self-ID packets during bus configuration. A node may require bus power for its link and other unit hardware associated with the node. The bus manager, having monitored all self-ID packets, can calculate the total bus power sourced by nodes on the bus, as well as the total bus power required by nodes. Based on this information, the bus manager can determine if sufficient power is available to support all nodes requiring bus power. Obviously, two possibilities exist and the actions that the bus manager must take are:

- Power required exceeds power available — the bus manager must notify its application by reporting an insufficient cable power event to the link layer, which is passed to the application at the bus manager node. The actual bus manager service used is an SB_EVENT indication with the insufficient bus power parameter set. The application at the bus management node must be prepared to handle this situation. The application is responsible for determining the appropriate action. Selected nodes may be chosen to receive power (via a link-on packet), or all nodes requiring bus power may be left powered off and the user notified.
- Power required is equal to or less than power available — in this instance, all nodes that have inactive link layer are powered via the link-on packet. The application uses the SB_CONTROL service to cause the link-on packet to be transmitted.
FireWire System Architecture

Format of the link-on packet is shown in Figure 19-1. The two most significant bits (01b) of the packet identify it as a link-on packet. The next six bits specify the node address that this packet is targeting. When the PHY receives this packet it initiates and passes an SB_EVENT indication to the node controller, which, in turn, switches bus power to the link.

*Figure 19-1: Format of the Link-On Packet Used to Apply Bus Power to a Node’s Link Layer*

<table>
<thead>
<tr>
<th>msb (transmitted first)</th>
</tr>
</thead>
<tbody>
<tr>
<td>01b phy_ID 000000h</td>
</tr>
<tr>
<td>logical inverse of first quadlet</td>
</tr>
<tr>
<td>lsb (transmitted last)</td>
</tr>
</tbody>
</table>

Power Management by IRM Node

The Isochronous Resource Manager (IRM) can provide a minimal level of power management by issuing link-on packets to these nodes that have an inactive link layer. The IRM recognizes those nodes that don’t have their link layer powered by observing the self-ID packets that have the “L” bit cleared (0). Note that the IRM need not verify that sufficient power is available before issuing the link-on packet.

The Topology Map

Knowledge of the bus topology can be used to optimize serial bus performance. This can be accomplished by:

- Reconfiguring the topology to reduce the number of cable hops. The specification doesn’t say how this should be accomplished, but user intervention is clearly required. One can envision an elaborate animated graphic to guide the user in reconfiguring the connections, or the entire issue may simply be ignored. We’ll see!
- Reconfiguring the topology so that devices with the same speed capability are arranged adjacent to each other. Once again the user must assist.
Chapter 19: Bus Manager

All bus manager capable nodes capture self-ID packets as they are sent by each node during bus configuration. Once a node is selected as bus manager, it constructs a topology map and makes it available to all nodes. (The topology map format is shown in Figure 19-2 on page 347.) The bus manager must also perform consistency checks to ensure that the total number of ports connected to parents equals the number of ports connected to children. If the self-ID information received is inconsistent, then the length field must be cleared to zero and the topology error reported to the application via an SB_EVENT indication.

Figure 19-2: Topology Map Format

<table>
<thead>
<tr>
<th>length</th>
<th>CRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>generation_number</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
<tr>
<td>node_count</td>
<td>self_id_count</td>
</tr>
<tr>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>self_id_packet [0]</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>self_id_packet [self_id_count - 1]</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
</tr>
</tbody>
</table>

Accessing the Topology Map

An application associated with a given node that requires information from the topology map registers must perform the access using the following procedures:

1. Read the length field at offset 1000h within the initial units’ address space of the bus manager node. (Location of the bus manager is found by reading the BUS_MANAGER_ID register located at the isochronous resource manager node.) If the field contains a value of zero, then the topology is invalid. Otherwise, the topology information can be read.
2. Read the quadlet entries of interest.
20 Bus Management Services

The Previous Chapter

In the previous chapter, the bus manager function was described including topology map generation and access, speed map generation and access, and power management functions.

This Chapter

This chapter discusses the bus management services that are used by the bus manager and isochronous resource manager to perform their bus management roles.

The Next Chapter

The next chapter discusses the CSR registers defined by the ISO/IEC 13213 specification with particular focus on the registers that are required by the 1394-1995 and 1394a specifications.

Overview

A variety of bus management activities must be performed by the isochronous resource manager and the bus manager nodes as described in the two previous chapters. The local application at the node must be designed to support bus management activities if it is bus manager or isochronous manager capable. Bus management services are defined by the specification which provides the interface between the application and the bus management layer. The bus management layer passes messages and status information to the application regarding the state of the bus or management layer itself. The application also uses the interface to direct the bus management layer to take action regarding certain bus management issues. Figure 20-1 on page 352 illustrates the relationship between the bus management layer and the rest of the node.
Figure 20-1: Relationship Between Bus Management Layer and the Rest of the Node
Chapter 20: Bus Management Services

Three services are defined for the serial bus management layer:

- Serial Bus control request (SB_CONTROL.request)
- Serial Bus control confirmation (SB_CONTROL.confirmation)
- Serial Bus event indication (SB_EVENT.indication)

The Serial Bus control services are used by the application to direct the bus management layer to take some action (the request) and once the bus management layer has performed the requested operation’s verification is sent back to the application (the confirmation). Status information is passed to the application by the bus management layer using the Serial Bus event indication. Each service is defined in the following sections.

Serial Bus Control Requests

The Serial Bus control request service can be used not only to specify that some action be taken related to serial bus management but also to request status information. The following list specifies the actions that can be requested by an application using the Serial Bus control request:

- Reset the bus
- Initialize the node
- Transmit a Link-on packet
- Present Status
- Transmit a PHY configuration packet

Each action is detailed below.

Bus Reset Control Request

When this request is issued by the application layer, the PHY layer is directed to signal a bus reset and initialize itself. The reset request also directs the link and transaction layer to discard all pending transactions and subactions.

The bandwidth set-aside for isochronous transactions is also passed by the bus manager application to specify the amount of bus bandwidth that should be reserved for asynchronous transactions. The bus manager application (or isochronous resource manager if no bus manager exists on the bus) specifies the number of allocation units to be subtracted from the bus BANDWIDTH_AVAILABLE register that resides within the link layer of the isochronous resource manager. The bus manager must then update the
BANDWIDTH_AVAILABLE register to reflect the bandwidth that remains for isochronous transfers. Note that this requires generation of a lock compare and swap transaction targeting the BANDWIDTH_AVAILABLE register.

Initialize Control Request

The initialize control request is used to reinitialize the node and prepare it to transmit and receive packets. Specifically, this request directs the link and transaction layers to discard all pending transactions and subactions and enable the link to receive packets and also enable the transaction layer to accept transaction requests from the application.

Link-On Control Request

This request is made only by the bus manager (required) or isochronous resource manager (optional) applications. This request directs the PHY to generate a Link-on packet to notify the target node to attempt the application of power to its link layer controller. This service also passes the physical ID of the node to whichever power is to be applied.

Present Status

This request is issued to request status information be returned to the application from the bus management layer. The SB_CONTROL.confirmation returns the status information as listed on “Serial Bus Control Confirmations” on page 356.

PHY Configuration Request

The bus manager or isochronous resource manager node are the only nodes that issue the PHY configuration request. This request causes the generation of a PHY configuration packet or one of the extended PHY packets.
Set Force Root and Set Gap Count

This broadcast packet provides the ability to change the gap count variable and to force the Root Hold Off (RHB) in the specified PHY, while forcing all other nodes to clear the RHB. The parameters passed with this request include:

- **Set force root** — this flag when set indicates that the “physical ID” parameter is valid and that the “R” bit in the PHY configuration packet must be set.
- **Physical ID** — This parameter specifies the target node that must set its force root bit, making it the root following the next bus reset. This parameter is valid if the “set force root” parameter is set.
- **Set gap count** — this parameter when set indicates that the gap count field contains a valid gap count value and that the “T” bit in the PHY configuration packet must be set.
- **Gap count** — this parameter specifies the value of the gap count field that will be sent via the PHY configuration packet. This parameter is valid when the “set gap count” parameter is set.

Extended PHY Packets

The 1394a supplement defines extended PHY packets that can be generated via the PHY configuration request. The extended packets use the PHY configuration packet format with the gap_count field specifying the extended packet type. When the PHY configuration request is issued with the “set force” and “set gap count” parameters both cleared, the gap count parameter specifies the extended packet type. Depending on the extended packet type, the physical ID parameter may define extended packet specific information. (See Chapter 10 for details regarding extended PHY packets. Parameter data are specified below for extended PHY configuration packet generation.

- **Set force root** — this parameter must be cleared.
- **Set gap count** — this parameter must be cleared.
- **Gap count** — this parameter defines one of the following types of extended PHY packets:
  - Ping packet
  - Remote access packet
  - Remote reply packet
  - Remote command packet
  - Remote confirmation packet
  - Resume packet
- **Physical ID** — physical ID of the target node.
The Previous Chapter

The previous chapter discussed the bus management services that are used by the bus manager and isochronous resource manager to perform their bus management roles.

This Chapter

This chapter discusses the CSR registers defined by the ISO/IEC 13213 specification with particular focus on the registers that are required by the 1394 specification.

The Next Chapter

The following chapter details the contents of configuration ROM required by the ISO/IEC 13213 specification. The serial bus also defines ROM entries that are required by some nodes, depending on the capabilities.

Overview

Firewire is based on the ISO/IEC 13213 specification, commonly referred to as the Control and Status Registers (CSR) Architecture for microcomputer buses. This specification defines a common set of core features that can be implemented by a variety of buses that adhere to this standard. A group of core registers support functions common to CSR architecture buses and provide standardized offset locations within the initial register address space where these registers can be accessed. The start address location of the initial register space is at offset FFFF F000 0000h (the top 256MB block of address space) from the beginning of the node’s address space as illustrated in Figure 21-1 on page 362. Note that the CSR architecture also defines configuration ROM, which is discussed in the following chapter.
The IEEE 1394 specification defines the subset of CSR architecture features that must be supported for serial bus compliance, and also defines specific serial bus dependent extensions to the CSRs.

Core Registers

Table 21-1 on page 363 lists the core CSR registers defined by the CSR architecture and specifies the location of the serial bus dependent registers. The CSR architecture registers required in serial bus nodes are shaded in Table 21-1. The lighter shaded entry indicates that the register is conditionally required. The definition of each register and related register fields are specified in the following sections.
Table 21-1: Core CSR Locations and Definition

<table>
<thead>
<tr>
<th>Offset (h)</th>
<th>Register Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>STATE_CLEAR</td>
<td>State &amp; control information</td>
</tr>
<tr>
<td>004</td>
<td>STATE_SET</td>
<td>Sets STATE_CLEAR bits</td>
</tr>
<tr>
<td>008</td>
<td>NODE_IDS</td>
<td>Specifies 16-bit node ID value</td>
</tr>
<tr>
<td>00C</td>
<td>RESET_START</td>
<td>Resets state of node</td>
</tr>
<tr>
<td>010-014</td>
<td>INDIRECT_ADDRESS, INDIRECT_DATA</td>
<td>Indirectly access ROMs &gt; 1KB</td>
</tr>
<tr>
<td>018-01C</td>
<td>SPLIT_TIMEOUT_HI, SPLIT_TIMEOUT_LO</td>
<td>Split-request timeout</td>
</tr>
<tr>
<td>020-02C</td>
<td>ARGUMENT_HI, ARGUMENT_LO, TEST_START, TEST_STATUS</td>
<td>Optional diagnostic-test interface</td>
</tr>
<tr>
<td>030-04C</td>
<td>UNITS_BASE, UNITS_BOUND, MEMORY_BASE, MEMORY_BOUND</td>
<td>Never implemented</td>
</tr>
<tr>
<td>050-054</td>
<td>INTERRUPT_TARGET, INTERRUPT_MASK</td>
<td>Optional broadcast/nodecast interrupt</td>
</tr>
<tr>
<td>058-07C</td>
<td>CLOCK_VALUE, CLOCK_TICK_PERIOD, CLOCK_STROBE_ARRIVED, CLOCK_INFO</td>
<td>Synchronized time-of-day value and control</td>
</tr>
<tr>
<td>080-0FC</td>
<td>MESSAGE_REQUEST, MESSAGE_RESPONSE</td>
<td>Optional message passing register</td>
</tr>
<tr>
<td>100-17C</td>
<td>RESERVED</td>
<td>Reserved for CSR architecture</td>
</tr>
<tr>
<td>180-1FC</td>
<td>ERROR_LOG_BUFFER</td>
<td>Reserved for Serial Bus</td>
</tr>
<tr>
<td>200-3FC</td>
<td>SERIAL BUS DEPENDENT</td>
<td>See Table 21-5 on page 376</td>
</tr>
</tbody>
</table>

- **Required**
- **Conditionally Required**
Effect of Reset on the CSRs

Three types of Reset are supported by the serial bus:

- **Power reset**—defined by the CSR architecture, it occurs when power is applied to the link, PHY, and bus management functions. This causes all CSR registers to return to their initial values. The PHY layer is also reset and a bus reset is initiated.
- **Command reset**—defined by the CSR architecture and caused by a write to the RESET_START register. This reset does not result in the PHY being reset nor does it cause a bus reset.
- **Bus reset**—defined by the IEEE 1394 specification and caused by the addition or removal of a node or by a change in the powered state of the PHY layer of a node.
- **Software initiated bus reset**—a bus reset caused by the local node application writing to either the IBR (initiated bus reset) or ISBR (initiated short bus reset) bit in the PHY register space.

A power reset forces all CSR registers to the initial values. When a command or bus reset occurs, CSR registers are sometimes also returned to their initial values, while in other instances particular fields may remain unchanged as discussed in the following sections.

State Register (State_Clear & State_Set)

The STATE register is defined by the CSR architecture and provides support for status and control features. Although these registers are defined as optional within the CSR architecture specification, they are required by the serial bus specification. The STATE_CLEAR register is used to clear state bits, while the STATE_SET register provides a way to set state bits.

The STATE register format is illustrated in Figure 21-2, and the definition and usage of each field is specified in Table 21-2 on page 365. Note also that Figure 21-2 on page 365 defines the values of the individual state bits following initialization and reset, and shows the read values returned and the effects of writes on each field. The bus_dependent field is defined by the IEEE 1394 specification. Its format is illustrated in Figure 21-3 on page 367 and field definitions are listed in Table 21-3 on page 368.
Chapter 21: CSR Architecture

State_Clear Register

Writing a value of one to a writable bit within the STATE_CLEAR register location forces that bit to be cleared to zero. Writing a zero to a bit position has no effect on the current value. This is implemented by ANDing the complement of the write-data value to the current state-bit value.

Figure 21-2: Format of the STATE Register

<table>
<thead>
<tr>
<th>Field Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit_depend</td>
</tr>
<tr>
<td>16</td>
</tr>
</tbody>
</table>

Table 21-2: Field Definitions for Register

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>unit_depend</td>
<td>Bits within this field are intended for use by the unit architecture. Definition of these bits, if defined, are provided by a unit architecture specification.</td>
</tr>
<tr>
<td>bus_dependent</td>
<td>These bits are defined by the appropriate bus standard. The IEEE 1394 specification defines usage of these bits. See Figure 21-3 on page 367 and Table 21-3 on page 368 for details.</td>
</tr>
<tr>
<td>lost</td>
<td>This bit must be implemented by serial bus nodes. This bit is set when a power reset occurs or the node has transitioned to the dead state (due to an error). This bit is not directly affected by a bus reset. Software is expected to clear the lost bit after the node has been initialized and the I/O driver has been notified of the reset or a fatal error in the event of a node transition to the dead state.</td>
</tr>
<tr>
<td>dreq</td>
<td>Dreq must be implemented by serial bus nodes that are capable of initiating transaction requests. The dreq (disable request) bit is intended to be set by software to disable request generation from unreliable nodes. Nodes may provide a “back door” access to this bit so that the bit may be cleared by special-purpose processors (e.g. via a remote diagnostic interface). Nodes that cannot initiate transaction requests but can respond to such requests must have this bit permanently set (1).</td>
</tr>
<tr>
<td>res</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
The Previous Chapter

The previous chapter discussed the CSR registers defined by the ISO 13213 specification with particular focus on the registers that are required by the 1394 specification.

This Chapter

This chapter introduces PHY register maps and port registers for the 1394-1995 specification and for the 1394a supplement.

The Next Chapter

The next chapter details the contents of configuration ROM required by the ISO/IEC 13213 specification. The serial bus also defines ROM entries that are required by some nodes, depending on the capabilities.

Overview

Each 1394 PHY provides the interface to the bus and performs key functions in the communications process. These functions include:

- Bus configuration
- Arbitrating for control of the 1394 bus
- Repeating transactions to other ports
- Performing NRZ encoding/decoding
- Performing data strobe encoding/decoding
- Speed signaling and detecting transfer speed
- Detecting device attachment/detachment

The PHY registers support the functions performed by the PHY layer. These registers are mapped as offsets within the PHY and are not mapped into the 1394 node address space. The PHY registers can be read from or written to by the application residing at node and can be read by a remote node using a remote access packet. Additionally, another node can use the PHY configura-
FireWire System Architecture

tion and link-on packets to change certain bits within the PHY registers of other nodes. This function is reserved for the node that performs bus management functions. These packets affect register fields related to force root, gap count, and link power features.

The section entitled, “1394-1995 PHY Register Map” details the PHY register map used by 1394-1995 compliant nodes. The 1394a supplement defines an extended PHY register map format that contains additional information needed to support new features, and is discussed in the section entitled, “1394a PHY Register Map” on page 398.

1394-1995 PHY Register Map

Figure 22-1 illustrates the PHY register map for the 1995 version of the specification. The register map contains global information that pertains to the entire node, as well as port specific registers that reflect the state and condition of each port interface. A description of each field within the PHY register map is presented in Table 22-1.

Figure 22-1: 1394-1995 PHY Register Format

<table>
<thead>
<tr>
<th>0000</th>
<th>Physical_ID</th>
<th>R</th>
<th>PS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001</td>
<td>RHB</td>
<td>IBR</td>
<td>Gap_count</td>
</tr>
<tr>
<td>0010</td>
<td>SPD</td>
<td>E</td>
<td>#ports</td>
</tr>
<tr>
<td>0011</td>
<td>AStat0</td>
<td>BStat0</td>
<td>Ch0</td>
</tr>
<tr>
<td>0100</td>
<td>AStat1</td>
<td>BStat1</td>
<td>Ch1</td>
</tr>
<tr>
<td>0101</td>
<td>AStat2</td>
<td>BStat2</td>
<td>Ch2</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>#Ports + 0011</td>
<td>AStat&lt;nports&gt;</td>
<td>BStat&lt;nport&gt;</td>
<td>Ch&lt;nports&gt;</td>
</tr>
<tr>
<td>#Ports + 0100</td>
<td>ENV</td>
<td>Register_count</td>
<td></td>
</tr>
<tr>
<td>#Ports + 0101</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### Table 22-1: Description of the PHY Register Map For 1394-1995

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bits)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical_ID</td>
<td>6</td>
<td>This field is updated during bus configuration (self-ID) to reflect the node ID of this device.</td>
</tr>
<tr>
<td>R</td>
<td>1</td>
<td>Root — Designates whether this node is the root node. When set to one this node is the root.</td>
</tr>
<tr>
<td>PS</td>
<td>1</td>
<td>Power status — The PHY sets this bit when it detects valid power (i.e. in the range of 7.5 - 33vdc).</td>
</tr>
<tr>
<td>RHB</td>
<td>1</td>
<td>Root Hold Off Bit — This field is set when the PHY detects a PHY Configuration packet whose Root_ID field matches the Physical_ID of this node and whose R bit is set. When RHB is set this node delays its participation in the tree-ID process.</td>
</tr>
<tr>
<td>IBR</td>
<td>1</td>
<td>Initiate Bus Reset — When set to one this bit causes the PHY to signal bus reset immediately. Reset is asserted for 166 microseconds after which the IBR bit is self cleared.</td>
</tr>
<tr>
<td>Gap_count</td>
<td>6</td>
<td>This register contains an initial default value of 63 following reset. This value can be updated later by the bus manager or isochronous manager nodes via the PHY configuration packet. The gap_count field value of the PHY configuration packet updates the PHY gap_count register when the “T” bit is set.</td>
</tr>
<tr>
<td>SPD</td>
<td>2</td>
<td>Indicates the top speed that this PHY can accept and transmit packets. 00=100Mb/s 01=200Mb/s 10=400Mb/s 11=Reserved</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>Enhanced bit: 1=enhanced register map is used (registers beyond #ports+0100 are defined.</td>
</tr>
<tr>
<td>#Ports</td>
<td>5</td>
<td>The number of ports supported by this PHY. This field determines the number of port status registers that directly follow this register field.</td>
</tr>
</tbody>
</table>
### Table 22-1: Description of the PHY Register Map For 1394-1995

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bits)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AStat(n)</td>
<td>2</td>
<td>TPA line state on port (n) (0-max Port#) encoded as follows: (11=ZZ), (01=1), (10=0), (00=invalid)</td>
</tr>
<tr>
<td>BStat(n)</td>
<td>2</td>
<td>TPB line state on port (n) (0-max Port#) encoded as follows: (11=ZZ), (01=1), (10=0), (00=invalid)</td>
</tr>
<tr>
<td>Ch(n)</td>
<td>1</td>
<td>If Ch(n)=1, port (n) is a child. If Ch(n)=0, port (n) is a parent.</td>
</tr>
<tr>
<td>Con(n)</td>
<td>1</td>
<td>If Con(n)=1, port (n) is connected. If Con(n)=0, port (n) is disconnected.</td>
</tr>
<tr>
<td>ENV</td>
<td>2</td>
<td>Used with enhanced register. Indicates the type of environment: (00=\text{backplane}), (01=\text{cable}), (10\ &amp; \ 11=\text{reserved})</td>
</tr>
<tr>
<td>Reg_count</td>
<td>6</td>
<td>Defines number of registers that follow in the enhanced space.</td>
</tr>
</tbody>
</table>
Port Status Registers

The port status registers contain information regarding the connection status of each port and, if a node is attached, whether the port connects to a child or to a parent node. This information is delivered as part of the self_ID packet as port specific information.

PHY Configuration Packet

Figure 22-2 illustrates the PHY configuration packet contents. Two PHY register fields may be affected by the configuration packet:

1. Root Hold Off Bit (RHB) field
2. Gap_count field

Root Hold Off

When a PHY configuration packet is broadcast with the R bit set, then the node must compare its physical_ID to the root_ID field in the configuration packet. If the two values match, then this root must set its RHB field. If the two values do not match, then the PHY must clear the RHB. When RHB is set, this PHY will delay its participation in the tree-ID process for approximately 167µs. This ensures that this node will become the root during the next tree-ID process. Refer to “Force Root Delay” on page 299 for details regarding the tree_ID process and the force root feature.

Gap Count Optimization

The gap count field configures the gap timing employed by this PHY (e.g. when arbitrating for control of the bus and when detecting time-out conditions). The bus manager or isochronous resource manager can optimize bus performance by tuning the gap count value. The default gap count of 63 can be shortened to reduce idle time between packets. When the PHY configuration packet is delivered, all nodes check the “T” bit. If set, the “T” bit indicates that the gap_count value contained in the configuration packet should replace the current value within the PHY register’s gap_count field. Since the PHY configuration packet is a broadcast packet, all nodes will update their gap_count register field to the same value.
The Previous Chapter

The previous chapter discussed the CSR registers defined by the IEEE 1212 specification with particular focus on the registers that are required by the 1394 specification. Additional bus-specific registers are also defined by the 1394 specification and are discussed.

This Chapter

This chapter details the contents of configuration ROM required by the ISO/IEC 13213 specification. The serial bus also defines ROM entries that are required by some nodes, depending on the capabilities.

The Next Chapter

The next chapter provides a brief introduction to the power management environment introduced by the 1394a specification. The chapter introduces the three documents that further define the power management specification: Cable Power Distribution, Suspend/Resume Mechanisms, and Power State Management.

Overview

IEEE 1394 serial devices must include a ROM directory structure that provides critical information needed to configure and diagnose problems associated with the device. Information included within the ROM includes information for:

- identifying the software driver for this device
- identifying diagnostic software
- specifying bus-related capabilities of the device (e.g., whether it is bus manager capable)
- specifying optional module, node, and unit characteristics and parameters
FireWire System Architecture

The specification defines two ROM formats: minimal and general. The minimal format only identifies the company that manufactured the device, but it may also include vendor-defined data structures. The general ROM format defines a bus information block and root directory containing entries that may specify pointers to other directories and data structures.

Minimal ROM Format

Figure 23-1 illustrates the minimal ROM format that consists only of a 24-bit Vendor-ID value. The most significant 8 bits contain a value of 01h, which identifies the ROM format as minimal. Any other value will be interpreted as a general ROM format. The vendor may optionally define and implement other ROM entries. These additional entries are entirely vendor-defined and not a part of the CSR architecture or the serial bus standard and can only be interpreted by vendor software.

Figure 23-1: Minimal ROM Format

<table>
<thead>
<tr>
<th>01h</th>
<th>vendor_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>24</td>
</tr>
</tbody>
</table>

General ROM Format

Major entries within the general ROM format consist of a bus information block and root directory. The bus information block specifies a variety of bus-related capabilities, while the root directory provides values that identify the software driver and diagnostic software along with optional pointers to other directories and data structures. Figure 23-2 illustrates the general ROM format, with the entries required by the CSR architecture shaded. Note that these shaded structures always start at the same address locations within ROM. Whether the other directories and data structures are present is bus- and vendor-dependent.
Chapter 23: Configuration ROM

Header Information

The first quadlet of the general ROM format consists of three fields:

- info_length
- crc_length
- rom_crc_value

Info_Length

This field specifies the length of the bus_info_block field in quadlets. This value must be a value greater than 01h so that software can correctly identify the ROM format as "general" rather than "minimal."
FireWire System Architecture

**CRC_Length**

The crc_length field specifies that the total length of the general ROM quadlets are covered by the crc value. The field size of the crc_length values restricts the maximum number of quadlets that can be covered by the crc_value to 255 (1020 bytes). Note that these values cover the entire ROM up to the maximum size. However, other data structures within the general ROM also contain crc values that can be used to isolate an error that has been detected within the ROM. If the ROM is larger than the maximum size covered by the ROM_crc_value, then CRC checks should be performed on directories not covered by the ROM crc.

**CRC_Value**

Software performs a crc check on two byte groups that are covered by the crc. The calculated crc matches the crc_value when no errors are detected.

**Bus_Info_Block (1394-1995)**

The Bus_Info_Block provides critical information about bus related capabilities of this node. Only the bus_name field (the first quadlet) is specifically defined by the CSR architecture. The remainder of the Bus_Info_Block is defined by the serial bus standard. The format of the Bus_Info_Block content and format is illustrated in Figure 23-3. See page 415 for changes to the bus information block introduced by the 1394a supplement.
Chapter 23: Configuration ROM

Bus_Name Field

The CSR architecture defines the bus_name field that identifies the bus that this node supports based on four ASCII characters that represent the IEEE PAR number assigned to the corresponding bus standard. The serial bus specification defines this quadlet as “1394.”

Bus Characteristics Fields

A variety of bus characteristics associated with this node are defined in the second quadlet of the Bus_Info_Block. The fields illustrated in the second quadlet of Figure 23-3 define the following node characteristics:

- **irmc** (isochronous resource manager capable) — When set (1) this node is isochronous resource manager capable, otherwise the bit must be cleared (0).
- **cmc** (cycle master capable) — When set (1), this node is cycle master capable, otherwise the bit must be cleared (0).
- **isc** (isochronous capable) — When set (1), this node supports isochronous transfers, otherwise the bit must be cleared (0).
- **bmc** (bus manager capable) — When set (1), this node is bus manager capable, otherwise the bit must be cleared (0).
- **cyc_clk_acc** (cycle clock accuracy) — This 8-bit field contains a value that defines the accuracy of this node’s cycle master clock in parts per million. This field is only valid for nodes that also have the **cmc** bit set. Valid values are between zero and 100d. If **cmc** is cleared, then this field must contain all ones.
- **max_rec** (maximum data record size) — This 4-bit field indirectly specifies the maximum data payload size of asynchronous write and asynchronous stream packets that this node is capable of accepting. The max_rec value is used to calculate the maximum data payload value, which is an even power of two \(\text{max packet size} = 2^{\text{max_rec}+1}\). The valid max_rec values and corresponding maximum data payload sizes are listed in Table 23-1. The shaded table entries reflect new max_rec values that correspond with the faster transmission speeds defined by the 1394a supplement. Note that these faster speeds and maximum payload sizes are currently not supported.
The Previous Chapter

The previous chapter detailed the contents of configuration ROM required by the ISO/IEC 13213 specification. The serial bus also defines ROM entries that are required by some nodes, depending on the capabilities.

This Chapter

This chapter provides a brief introduction to the power management environment introduced by the 1394a specification. The chapter introduces the three documents that further define the power management specification: Cable Power Distribution, Suspend/Resume Mechanisms, and Power State Management.

The Next Chapter

The next chapter discusses power distribution in the cable environment. It discusses the four power type designations for nodes: power providers, alternate power providers, power consumers, and self-powered devices. Details regarding the power implementation of nodes are also included.

Overview

The 1394-1995 specification defines a variety of power-related issues that have been discussed in previous chapters. The 1394a supplement provides additional definition and capability regarding generation, distribution, and management of power in the 1394 environment. These additions are the focus of this chapter. Regrettably, the 1394a supplement and the associated Power Specification were
FireWire System Architecture

still in development at the time of this writing. The reader is strongly cautioned that some of the information contained in this chapter will likely change before final approval by the 1394 Trade Association and additional information may be added. Subsequent editions of the book will include information from the final specification.

Review of 1394-1995 Power-Related Issues

The 1394-1995 specification includes a variety of features related to powering 1394 devices. In general, the specification allows nodes to be powered either by their own local power supply or from the cable. However, it also requires that the physical layer of each node must be capable of repeating serial bus traffic whether or not local power is available. This means that all nodes must be able to power their PHY layer interface with cable power, in the event that local power is off. Additionally, the specification requires that there is sufficient cable power available to power all nodes.

Other power-related requirements include:

- A single connector type is defined that includes VP and VG pins.
- Nodes may (not required) supply unregulated power to the bus. (from 8vdc - 40vdc).
- Nodes must isolate cable power from local power.
- Nodes may consume no more than 1 w of power from the bus after reset. Additional power may be consumed if sufficient bus power is available. This is under the control of the node that provides power management support.
- All nodes must report their power class in the self-ID packet during bus configuration. If the total power required by the node (including power required by internal units) exceeds the amount that can be reported in the self-ID packet (10W total), it must report the additional power that it requires in configuration ROM.
- The bus manager node must calculate the total power available on the bus and determine if sufficient power is available to power all nodes that need cable power. If sufficient power is available, the bus manager must generate a link-on packet for each device that requires cable power.
- In the absence of a bus manager node, the isochronous resource manager node is responsible for issuing the link-on packets to apply power to nodes that require bus power. Note that the isochronous resource manager is not required to validate power availability prior to sending the link-on packets.
Chapter 24: Introduction to Power Management

Goals of the 1394a Power Extensions

Members of the personal computer industry involved in implementing the 1394 serial bus have concluded that the 1995 version of the specification requires additional clarification and enhancement. To this end, the 1394 Trade Association has drafted a three part specification to further define power-related issues for the 1394 serial bus. These documents contain the following information:

- Part 1 — defines power distribution on the serial bus, including the voltage levels, types of power providers, power consumers, self-powered devices, and power down behavior.
- Part 2 — defines power states, CSR registers, and configuration ROM entries used to control and manage power.
- Part 3 — defines power conservation mechanisms (suspend and resume) for implementing low power bus states. Note, however, that the suspend and resume features were still being debated when this book was published, and have not been included here. Check MindShare’s web site for later information.

The state of the power management documentation at the time of writing was:

Part 1: Cable Power Distribution (Revision 0.93)

Part 2: Suspend/Resume (Revision 0.73)

Part 3: Power State Management (Revision 0.71)
The Previous Chapter

The previous chapter provided a brief introduction to the power management environment introduced by the 1394a specification. It also discussed the state of the power specifications at the time of writing: Cable Power Distribution, Suspend/Resume Mechanisms, and Power State Management.

This Chapter

This chapter discusses power distribution in the cable environment. It discusses the four power type designations for nodes: power providers, alternate power providers, power consumers, and self-powered devices. Details regarding the power implementation of nodes is also included.

The Next Chapter

Next, the suspend and resume mechanism is defined. This capability allows the PHY layer within a node to enter a low power state under software control (either local node software or from another node). The mechanisms implemented for suspend and resume are detailed including: command and confirmation packets, suspend initiator actions, suspend target actions, and related suspend and resume signaling. The impact on PHY and port register definition is also discussed.

Power Distribution

The power distribution document provides additional definition for node power classes and introduces new terminology to describe nodes that supply power to the bus. In addition it defines the power functionality of each node in terms of four power configuration groups:
FireWire System Architecture

1. Power Providers
2. Alternate Power Providers
3. Power Consumers
4. Self-Powered nodes

Each of these configuration groups is described in the following sections.

**Power Class Codes**

The power class code definition for power providers, power consumers, and self-powered nodes is listed in Table 25-1 on page 432. Note that the class code may vary depending on whether the node is a single- or multi-port implementation. The definition of each code value is stated in Table 25-2 on page 432.

*Table 25-1: Node Power Class Code Assignments For Each Power Configuration*

<table>
<thead>
<tr>
<th>Node Power Configuration</th>
<th>Self-ID Packet Power Class</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Single Port</td>
</tr>
<tr>
<td>Power Provider</td>
<td>1, 2, 3</td>
</tr>
<tr>
<td>Alternate Power Provider</td>
<td>4</td>
</tr>
<tr>
<td>Power Consumer</td>
<td>4, 6, 7</td>
</tr>
<tr>
<td>Self-Power</td>
<td>0</td>
</tr>
</tbody>
</table>

*Table 25-2: Definition of Power Class Values Within “Pwr” Field of Self-ID Packets*

<table>
<thead>
<tr>
<th>POWER_CLASS Code (binary)</th>
<th>Power Consumption and Source Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>Node does not require bus power nor repeat bus power.</td>
</tr>
<tr>
<td>001</td>
<td>Node is self-powered and provides 15W (minimum) to the bus.</td>
</tr>
<tr>
<td>010</td>
<td>Node is self-powered and provides 30W (minimum) to the bus.</td>
</tr>
<tr>
<td>011</td>
<td>Node is self-powered and provides 45W (minimum) to the bus.</td>
</tr>
</tbody>
</table>
Chapter 25: Cable Power Distribution

Table 25-2: Definition of Power Class Values Within “Pwr” Field of Self-ID Packets (Continued)

<table>
<thead>
<tr>
<th>POWER_CLASS Code (binary)</th>
<th>Power Consumption and Source Characteristics</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Node may be powered from the bus and is using up to 3W and no additional bus power is needed to enable the link.</td>
</tr>
<tr>
<td>101</td>
<td>Reserved for future implementations.</td>
</tr>
<tr>
<td>110</td>
<td>Node is powered from the bus and consumes 3W maximum. An additional 3W maximum is needed to enable the link.</td>
</tr>
<tr>
<td>111</td>
<td>Node is powered from the bus and consumes 3W maximum. An additional 7W maximum is needed to enable the link.</td>
</tr>
</tbody>
</table>

Power Providers

Nodes that source power to the bus are termed power providers. Two classes of power providers are defined by the Power specification:

- Power Providers
- Alternate Power Providers

As shown in Table 25-2 on page 432, the amount of power that a node provides to the cable can vary. A power provider may be a single- or multi-port implementation and must use the 6-pin port connectors.

The cable power requirements are specified in Table 25-3. Note that these values differ from the 1394-1995 values.

Table 25-3: Cable Power Requirements

<table>
<thead>
<tr>
<th>Condition</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum output current per port</td>
<td>1.5 amps</td>
</tr>
<tr>
<td>Minimum output voltage (power class 1,2&amp;3)</td>
<td>20 vdc</td>
</tr>
<tr>
<td>Minimum output voltage (all other classes)</td>
<td>8 vdc</td>
</tr>
<tr>
<td>Maximum output voltage</td>
<td>33 vdc</td>
</tr>
<tr>
<td>Maximum output ripple (1 kHz to 400 MHz)</td>
<td>100 mv peak-to-peak</td>
</tr>
</tbody>
</table>
Any power provider must place a diode in the output line going to each port as illustrated in Figure 25-1 on page 434, versus a single diode for all ports for the 1995 version of the specification. This prevents current from flowing from the cable to the power supply when the cable voltage is higher than the power supply’s voltage.

*Figure 25-1: Power Providers Must Place Diodes in Each Port VG Line*

Power Provider Classes

The minimum amount of power supplied by a power provider is 15, 30, or 45W and must declare a power class of 1, 2, or 3 via its self-ID packet. The unregulated output voltage at each port must be in the range from 20vdc to 33vdc under full load conditions. Figure 25-2 on page 435 illustrates the configuration of a power provider. Power providers always deliver bus power as long as their local power source (battery or AC) remains enabled. The bus power output is not disrupted due to any action on the bus.
A power provider should not use bus power for its PHY in the event that local power is off. When the primary system power is lost a power provider may trickle power its PHY from another power source provided by the system. In this event, multi-port power providers must continue to repeat serial bus traffic to preserve the bus topology. A power provider whose PHY is not powered cannot repeat bus traffic, and consequently will have the effect of segmenting the bus. When trickle powering its PHY, a power provider must report its power class as 000b.

Since the power provider must have diodes on each port, it is also unable to pass power. The diodes provide a means of creating power domains. Domains provide a low-cost solution for adding power incrementally to the cable.

Current limiting must also be employed by all power providers to meet regulatory requirements and must not exceed 1.5A.

Figure 25-2: Configuration of Power Provider
Suspend & Resume

The Previous Chapter
The previous chapter discussed power distribution in the cable environment. It discussed the four power type designations for nodes: power providers, alternate power providers, power consumers, and self-powered devices. Details regarding the power implementation of nodes was also included.

This Chapter
This chapter introduces the suspend and resume mechanisms. This capability allows the PHY layer within a node to enter a low power state under software control (either local node software or from another node). The mechanisms implemented for suspend and resume are detailed including: command and confirmation packets, suspend initiator actions, suspend target actions, and related suspend and resume signaling. The impact on PHY and port register definition is also discussed.

The Next Chapter
The next chapter describes the CSR registers and ROM entries that define power management capabilities and provide the mechanisms for controlling the power states of a node and of local units within a node.

Overview
Due to the incomplete status of the suspend/resume documentation, this chapter provides only an overview of the suspend and resume capabilities and does not attempt to detail specific consideration and corner conditions that exist.
The goal of suspend and resume is to allow a pair of attached PHY ports to enter a low power state in which recovery to full power and operation is possible. In this way, a segment of the bus may be placed into a low power state. When a port is suspended it is no longer able to receive or transmit packets. However, suspended ports can detect whether a node is attached or detached.

To help understand the suspend capabilities consider the following example. Figure 26-1 on page 446 illustrates a 1394a only topology (i.e. all nodes are 1394a compliant) that interfaces to a PCI bus. Note that the node at the PCI bus has transferred a suspend command packet that target port 2 within the VCR node. In response, the VCR transmits a confirmation packet to confirm that the suspend command has been accepted. The command packet identifies the suspend initiator (the port responsible for signaling suspend).

Refer to Figure 26-2. Immediately following the confirmation packet (after detecting the acknowledge gap), the suspend initiator (port 2 of the VCR node) signals suspend (TX_SUSPEND=00) to the node connected to port 2 (video camera), which detects the suspend (RX_SUSPEND=00). The video camera is
referred to as the suspend target. The node containing the suspend initiator also
signals data prefix to its other ports followed by a short bus reset. Bus reset is
then repeated to all nodes in the system except the node or nodes attached to
the suspend initiator port.

The suspend initiator and suspend target perform a handshake and both enter a
low power, or suspended state. When the tree ID process is performed follow-
ing the short bus reset, the VCR node will report its port 2 as inactive, making
all nodes attached to the inactive port invisible to the rest of the network.

*Figure 26-2: Actions Taken by the Suspend Initiator*

Once the bus is reset and the suspended port handshake completes, the new
topology eliminates the video camera. No packets are transferred to the sus-
pended ports and they are unable to transmit packets. (See Figure 26-3 on page
448).
FireWire System Architecture

If the video camera is activated by the user, it can signal resume causing a port event indication that can serve to notify the link and application. Further the video camera can signal the VCR of the event. The bus can again be reset to cause bus reconfiguration with the VCR and video camera ports once again active. Similarly, the PCI node may cause a resume operation by sending a resume command packet to the VCR. This would cause the VCR and video camera port to transition to the active state. Reset would be signaled and the bus reconfigured.

Figure 26-3: State of Network Following Bus Reset

Suspending a Port

An active port may enter a suspended state under a variety of conditions:

- When a suspend command packet is received. The port number is specified in the address field of a suspend command packet. The command may be issued by another node or by the local link layer.
- When the port detects suspend signaling (RX_SUSPEND) via its arbitration...
comparators.

- When the port resides within the same node as another port that has detected RX_SUSPEND.
- When the port receives disable notify (RX_DISABLE_NOTIFY).
- When TpBias is no longer detected.

**Suspending via the Suspend Command Packet**

Figure 26-4 on page 449 illustrates the format of the command packet that specifies “initiate suspend.” The port number field specifies the specific port within the target PHY that will become the suspend initiator. The target node responds to the command packet with a confirmation packet, illustrated in Figure 26-5. This packet contains the initiate suspend command to tie this confirmation to the previous command. The phy_ID field contains the phy ID of the confirming node.

*Figure 26-4: “Initiate Suspend” Command Packet Format*

<table>
<thead>
<tr>
<th>msb (transmitted first)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
</tr>
<tr>
<td>logical inverse of first quadlet</td>
</tr>
<tr>
<td>lsb (transmitted last)</td>
</tr>
</tbody>
</table>

*Figure 26-5: Suspend Confirmation Packet Format*

<table>
<thead>
<tr>
<th>msb (transmitted first)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00b</td>
</tr>
<tr>
<td>logical inverse of first quadlet</td>
</tr>
<tr>
<td>lsb (transmitted last)</td>
</tr>
</tbody>
</table>
The Previous Chapter

The previous chapter introduced the suspend and resume mechanisms. This capability allows the PHY layer within a node to enter a low power state under software control (either local node software or from another node). The mechanisms implemented for suspend and resume were detailed including: command and confirmation packets, suspend initiator actions, suspend target actions, and related suspend and resume signaling. The impact on PHY and port register definition were also discussed.

This Chapter

This chapter describes the CSR registers and ROM entries that define power management capabilities and provide the mechanisms for controlling the power states of a node and of local units within a node.

Power Management

Power management involves the control of the power states of individual nodes or units within nodes. Four power states are defined that permit control over the node and individual units. To support power management, additional CSR registers and ROM entries are also defined.

The goal of power management is to enable applications to control transitions in the power states, so that power can be managed efficiently. This capability includes the following:

- A mechanism to allow applications at one node to determine the power-related abilities of a remote node or functional units within a node, and to determine its current power state.
- A mechanism to permit a remote application to enable a power feature and control the power state.
FireWire System Architecture

- A mechanism to allow a remote node to notify the power manager node of changes in its power state that affect bus operation.
- A mechanism to notify a node that has entered a low power state that a remote event has occurred that should wake the node up.
- Ability of the power manager node to facilitate the actions and capabilities required by the power management model, to determine the abilities of a power provider node, and to control the level of power that a power provider supplies to the bus.

Note that some of the power management features are controlled directly by the power manager node (i.e. the bus manager node), while other management features at the functional unit level are implementation specific and are intended to be controlled by the related application.

Power States

Four power states provide the ability to control power within each node and a functional unit within nodes. The definition of the node power states are defined for global power management control, while unit power states are specific to a given functional unit.

Node Power States

Node power states provide the operational states of the PHY and link. Table 27-1 lists the node power states. New CSR registers are defined that control the transition between states. Each state is summarized below.

<table>
<thead>
<tr>
<th>Node Power State</th>
<th>Link Power</th>
<th>PHY Power</th>
<th>Node Context</th>
</tr>
</thead>
<tbody>
<tr>
<td>N0</td>
<td>On (or standby)</td>
<td>On</td>
<td>Preserved</td>
</tr>
<tr>
<td>N1</td>
<td>Off</td>
<td>On</td>
<td>Unspecified</td>
</tr>
<tr>
<td>N2</td>
<td>Off</td>
<td>Suspend</td>
<td>Unspecified</td>
</tr>
<tr>
<td>N3</td>
<td>Off</td>
<td>Off</td>
<td>Lost</td>
</tr>
</tbody>
</table>

Table 27-1: Node Power States
Chapter 27: Power State Management

Node Power State Zero (N0). This state represents the full-on power state in which the PHY and link (via a link-on packet) are both powered.

Note that N0 may include an optional standby feature, in which the link may be partially suspended and the transaction and higher layers may be in a fully suspended state, thus reducing power consumption. If a transaction targets a node that is in standby, the link must be able to decode the address and return an ACK_TARDY acknowledge code (See Table 8-15 on page 192). The link must also return to its fully functional state and initiate a wake up to the higher layers. Once the node has recovered from the standby condition, it can service the request when it is retried by the requesting node.

The amount of power consumed by the node in the N0 state is required to be less than the value reported in the self-ID packet. Actual power consumption may be reported in a new configuration ROM entry called the Node_Power_Level entry. There may exist multiple Node_Power_Level entries, one for each state.

Node Power State One (N1). The N1 state reflects the condition of the node prior to receiving the link-on packet. This state exists between bus reset and receipt of the link-on packet.

Node Power State Two (N2). N2 is the suspend state during which the link has been powered off and the PHY is in a low power condition. Consequently, the PHY is unable to perform any of its normal functions (i.e. it cannot repeat, receive, or transmit packets). The only action supported by the PHY in the N2 state is that it can detect a remote wakeup signal, causing it to return to the N0 state.

Node Power State Three (N3). N3 represents the full off condition in which the link and PHY are not powered. In this condition the node context is lost.

Unit Power States

Four power states are also defined for functional units within the node. These states are defined in Table 27-2. The unit power states have an implied relationship with an application or software driver that utilizes the functional unit.

The relationship between the node and unit power states must also be maintained to ensure sufficient power is available to support the level of power required by the unit. The node power state must always provide equal (same state value) or greater (lower state value) capability than the unit. For example,
FireWire System Architecture

if one or more units within the node are currently running at a power state of D1, it is the responsibility of the node to not place the node into a power state lower than N1 (i.e. any request to place the node into the N2 or N3 power state would only result in a transition to N1).

Table 27-2: Unit Power States

<table>
<thead>
<tr>
<th>Node Power State</th>
<th>Operational Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D0 (required)</td>
<td>Fully operational</td>
<td>All unit context is maintained and full functionality is available. This state is required by all units.</td>
</tr>
<tr>
<td>D1 (optional)</td>
<td>Unit dependent</td>
<td>Power consumption is this state is less than D0. The time required to transition from D1 to D0 is less than the time to transition from D0 to D2. Unit context is preserved, but some functionality may be lost.</td>
</tr>
<tr>
<td>D2 (optional)</td>
<td>Unit dependent</td>
<td>Power consumption is this state is less than D1. The time required to transition from D2 to D0 or D1 is less than the time required to transition from D3 to either D0 or D1. Unit context may not be preserved.</td>
</tr>
<tr>
<td>D3 (optional)</td>
<td>Not operational</td>
<td>No power is consumed by the unit and external power may be removed.</td>
</tr>
</tbody>
</table>
Chapter 27: Power State Management

New CSRs

A variety of new CSR registers are required to support the power management functions. Nodes that support the new power management capability must implement some of the power-related CSRs; these nodes include:

- Power management capable nodes
- Nodes that support power management or implement one or more units that support power management.
- Nodes that implement batteries
- Units that support power management
- Power providers

The power-related CSRs are mapped within a node’s initial units address space at location FFFF F001 0000h or higher. Two groups of registers are defined:

- node-specific CSRs — the start address of this register block is specified by the Node_Power_Management entry within configuration ROM. Each power-related CSR occupies a specific offset within the block as illustrated in Table 27-3.
- unit-specific CSRs — the location of the base address is specified by the node’s Unit_Power_Management entry within configuration ROM. Each power-related unit CSR occupies a specific offset, illustrated in Table 27-4.

Table 27-3: Node-Specific Power-Related CSRs

<table>
<thead>
<tr>
<th>Relative Offset</th>
<th>Name</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h</td>
<td>NODE_POWER_STATE</td>
<td>Mandatory</td>
<td>Reports the node’s power state.</td>
</tr>
<tr>
<td>04h</td>
<td>NODE_POWER_CONTROL</td>
<td>Mandatory</td>
<td>Permits the node’s power state to be managed.</td>
</tr>
<tr>
<td>08h</td>
<td>NOTIFICATION_ADDRESS</td>
<td>Optional</td>
<td>Destination address for power status change notification or request.</td>
</tr>
<tr>
<td>Ch</td>
<td>CABLE_POWER_SOURCE_STATE</td>
<td>Optional</td>
<td>Reports the node’s current power provider status.</td>
</tr>
<tr>
<td>10h</td>
<td>CABLE_POWER_SOURCE_CONTROL</td>
<td>Optional</td>
<td>Permits the node’s power provider status to change.</td>
</tr>
</tbody>
</table>
Index

**Numerics**
1394-1995 210, 279
1394a 211
1394a supplement 16, 20, 87
4-conductor cable 90
4-pin connector 85, 87
64-bit fixed addressing 28
6-conductor cable 89
6-pin connector 85, 86

**A**
acceleration control 159
ack_busy_A 251, 256
ack_busy_B 251, 257, 258
ack_busy_x 249
ack_data_err 260
ack_data_error 259
ack_type_err 260
ack_type_error 259
ack-accelerated arbitration 157
acknowledge accelerated arbitration 157
acknowledge codes 57, 72, 192, 248
acknowledge Gap 125
acknowledge packet 57, 191
acknowledge packet generation 72
acknowledge packet parity error 259, 261
acknowledge packet time-out 259
acknowledge time-out 260
active domains 452
address space 17, 28
ARB_RESET_GAP 257
arbitrated (short) bus reset 283
arbitrated bus reset 283
arbitration 54, 112
arbitration enable bit 148
arbitration enhancements 157
arbitration line states 112
arbitration receiver decoding 108
arbitration receivers 107
arbitration reset gap 126
arbitration signaling 142
arbitration timeout 304
asymmetric connection change 282
asynchronous arbitration 147
asynchronous bus bandwidth 40, 152
asynchronous packet 56
asynchronous request receipt 71
asynchronous response generation 73
asynchronous stream packet 174
asynchronous transaction summary 193
asynchronous transaction, example 60
asynchronous transactions 38, 40, 56
asynchronous transfers 30, 37
attachment/detachment 275
AX_ARB_STATE_TIME 275

**B**
bandwidth set aside 341, 353, 356
BANDWIDTH_AVAILABLE register 79, 336, 339, 350, 353, 384
BANDWIDTH_AVAILABLE register, accessing 340
Battery_Group entry 476
Battery_State entry 476
BATTERY_STATE register 471
bias change 282
bias handshake 451
bias voltage 98
broadcast interrupts 34
broadcast messages 34
broadcast packet 46
bus bandwidth 79
bus bandwidth allocation 39, 44
bus bandwidth available register 353
Bus bandwidth set-aside 353
bus bandwidth set-aside 341, 350
bus configuration 55
bus idle 100
bus information block 412
bus initialization 273
bus initialization overview 267
bus management 343
bus management layer 42, 44, 351
bus management overview 272
bus management services 344, 351
bus manager 44, 344, 353
Index

bus manager ID 356
bus manager, identifying 344
bus powered nodes 138
Bus reset 278
bus reset 110, 273, 274, 277, 353, 364
bus reset signaling 275
bus reset, effects of 277
bus reset, sources of 274
Bus_Info_Block 412
BUS_MANAGER_ID register 344, 347, 383
BUS_TIME register 329, 377
busy retry 248
BUSY_TIMEOUT registre 249
BUSY_TIMEOUT register 252, 382
Cable assembly 90, 91, 92
cable cross section, 4-conductor 90
cable cross section, 6-conductor 89
cable electrical characteristics 89
cable length 89
cable power 56, 133
Cable Power Distribution Specification (Revision 0.93) 429
cable, 4-conductor 90
cable, characteristic impedance 89
cable, signal velocity 89
CABLE_POWER_SOURCE_CONTROL Register 467
Cable_Power_Source_Level entry 475
Cable_Power_Source_State register 466
cycle master enable (cmstr) 368
cycle master ID 356
cycle master, enabling 329
cycle start jitter 154
cycle start packet 152, 330, 379
cycle start skew 152
cycle synchronization event 79
CYCLE_TIME register 161, 377
cyclic redundancy check 57
common mode signaling 96
communications model 38
company_ID value 423
compare and swap 338
concatenated transactions 159
configuration 23, 34, 110
configuration overview 266
configuration ROM 33, 409
confirmation packet 449
Confirmation service 45
connect change 455
connect detect 451, 453
connect_timeout 280
connection change 278, 282
connection detection 280
constant current source 121
Control and Status Registers (CSR) Architecture 361
CRC 57
CRC checks 40
CRC error 259
CSR 33
CSR architecture - 1394 specific 376
CSR architecture 25, 33, 46, 361
CSR register summary - 1394 specific 376
CSR register summary - Core regs 363
CSRs, power-related 463
cycle master 44, 329
cycle master enable (cmstr) 368
cycle master ID 356
cycle master, enabling 329
cycle start jitter 154
cycle start packet 152, 330, 379
cycle start skew 152
cycle synchronization event 79
CYCLE_TIME register 161, 377
Cyclic redundancy check 57
<table>
<thead>
<tr>
<th>Index</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D</strong></td>
</tr>
<tr>
<td>data drivers 106</td>
</tr>
<tr>
<td>data end 114</td>
</tr>
<tr>
<td>data prefix signaling 114</td>
</tr>
<tr>
<td>DATA_PREFIX 144</td>
</tr>
<tr>
<td>data-strobe encoding 124</td>
</tr>
<tr>
<td>data-strobe encoding 39, 122</td>
</tr>
<tr>
<td>debounce time-out 280</td>
</tr>
<tr>
<td>debouncing connection detect 280</td>
</tr>
<tr>
<td>definition 446, 451</td>
</tr>
<tr>
<td>device bay 24, 93</td>
</tr>
<tr>
<td>Device Bay specification 20</td>
</tr>
<tr>
<td>differential input voltages 104</td>
</tr>
<tr>
<td>differential output voltages 104</td>
</tr>
<tr>
<td>differential signaling 96</td>
</tr>
<tr>
<td>disable port command 452</td>
</tr>
<tr>
<td>disabled change 455</td>
</tr>
<tr>
<td>dominance rule for one 108, 109</td>
</tr>
<tr>
<td>dribble bits 116</td>
</tr>
<tr>
<td>dual phase retry 251</td>
</tr>
<tr>
<td>dual phase retry state machine 252, 255</td>
</tr>
<tr>
<td><strong>E</strong></td>
</tr>
<tr>
<td>error detection 18</td>
</tr>
<tr>
<td>error handling 259, 292</td>
</tr>
<tr>
<td>Example asynchronous request 69</td>
</tr>
<tr>
<td>extended PHY packets 214, 355</td>
</tr>
<tr>
<td>extended type field 186</td>
</tr>
<tr>
<td><strong>F</strong></td>
</tr>
<tr>
<td>fair arbitration 18</td>
</tr>
<tr>
<td>fairness interval 147, 148</td>
</tr>
<tr>
<td>FAIRNESS_BUDGET register 161</td>
</tr>
<tr>
<td>fault 455</td>
</tr>
<tr>
<td>features of 1394 17</td>
</tr>
<tr>
<td>fly-by arbitration 157, 158</td>
</tr>
<tr>
<td>force root 213, 298</td>
</tr>
<tr>
<td>force root delay 299</td>
</tr>
<tr>
<td>fully managed bus 343</td>
</tr>
<tr>
<td>Futurebus+ 25</td>
</tr>
<tr>
<td><strong>G</strong></td>
</tr>
<tr>
<td>gap count 348</td>
</tr>
<tr>
<td>gap count field in PHY registers 278</td>
</tr>
<tr>
<td>gap count optimization 348, 397</td>
</tr>
<tr>
<td>gap timing 127</td>
</tr>
<tr>
<td>gap_count 278, 356</td>
</tr>
<tr>
<td>gaps 125</td>
</tr>
<tr>
<td>general ROM format 33, 410</td>
</tr>
<tr>
<td><strong>H</strong></td>
</tr>
<tr>
<td>hot plug connector 87</td>
</tr>
<tr>
<td><strong>I</strong></td>
</tr>
<tr>
<td>IBR (initiate bus reset) 275</td>
</tr>
<tr>
<td>IBR bit 278</td>
</tr>
<tr>
<td>IDENT_DONE 311</td>
</tr>
<tr>
<td>identification done (IDENT_DONE) 311</td>
</tr>
<tr>
<td>IEEE 1212 2, 13, 19, 20, 24, 37</td>
</tr>
<tr>
<td>IEEE 1394 25</td>
</tr>
<tr>
<td>IEEE 1394.B specification 20</td>
</tr>
<tr>
<td>IEEE 1394-1995 specification 20</td>
</tr>
<tr>
<td>IEEE 1596 25</td>
</tr>
<tr>
<td>IEEE896 25</td>
</tr>
<tr>
<td>immediate arbitration service 150</td>
</tr>
<tr>
<td>inactive domain 451</td>
</tr>
<tr>
<td>Indication service 45</td>
</tr>
<tr>
<td>initial memory space 28</td>
</tr>
<tr>
<td>initial register space 28</td>
</tr>
<tr>
<td>initial unit address space 388</td>
</tr>
<tr>
<td>initial units space 28</td>
</tr>
<tr>
<td>initialization state 273</td>
</tr>
<tr>
<td>interrupt broadcast 34, 375</td>
</tr>
<tr>
<td>interrupt enable 456</td>
</tr>
<tr>
<td>ISBR bit 278</td>
</tr>
<tr>
<td>ISO/IEC 13213 2, 13, 19, 20, 24, 25, 37, 361</td>
</tr>
<tr>
<td>isochronous arbitration 150</td>
</tr>
<tr>
<td>isochronous bus bandwidth 152</td>
</tr>
<tr>
<td>isochronous data packet size 201</td>
</tr>
<tr>
<td>isochronous gap 125, 141, 150</td>
</tr>
<tr>
<td>isochronous packet 46, 58</td>
</tr>
<tr>
<td>isochronous resource manager 44, 79, 333, 353, 356</td>
</tr>
<tr>
<td>isochronous resource manager (IRM) 346</td>
</tr>
<tr>
<td>isochronous resource manager, enabling 335</td>
</tr>
<tr>
<td>isochronous resource manager, identifying 334</td>
</tr>
</tbody>
</table>
Index

isochronous resource manager, power management 342
isochronous resource manager, requirements 335
isochronous resources, reallocation of 342
isochronous stream packet 199
isochronous transaction layer services 80
isochronous transaction summary 202
isochronous transaction, example 63
isochronous transactions 39, 44
isochronous transfers 15, 32, 38, 41, 78

K
key_type 418
key_value 418

L
line state decode 108
line states 105
link layer 42, 47
Link/PHY interface 222
Link/PHY interface, PHY reg access 241
Link/PHY interface, PHY status 238
Link/PHY interface, receiving packets 236
linkoff 368
link-on packet 205, 212, 346, 354
lock packet 183
lock request packet 184
lock response packet 189
lock transactions 183
lock types 186
lock, compare and swap 338
locked transaction 30, 31
long reset 279
looped topology 304
looped topology detection 304
lost 365

M
MAINT_CONTROL register 386
MAINTUTILITY register 386
make first/break last pins 87
MAX_ARB_STATE_TIME 274, 283
MAX_DATA_TIME 357
max_rec 413
maximum packet size 40
message broadcasts 34
minimal ROM format 33
module 25
Module_Vendor_Id entry 419
multiple buses 17
multi-speed concatenation 159

N
node 25
node attachment detect 97
node attachment/detachment 275
node state 366
Node_Capabilities entry 419
NODE_IDS register 369
NODEPOWER_CONTROL register 464
Node_Power_Directory entry 473
Node_Power_Level entry 474
Node_Power_Management Entry 475
Node_Unique_Id 414
Node_Unique_Id entry 419, 420
node_vendor_id 414
nodecast 375
non-return to zero 56, 123
NOTIFICATION_ADDRESS register 466
NRZ 56, 123
null isochronous packet 201

O
OHCI specification 20
Organizationally Unique Identifier (OUI) 423

P
packet acknowledgment 52
packet error handling 259
packet errors 259
packet size 40
packet transmission errors 259
Parent_Notify 286
partially managed bus 343
peer-to-peer transactions 17, 24
PHY configuration packet 205, 213, 354, 397
Index

PHY packet format 206
PHY register map - 1394a 398
PHY register map - 1995 394
PHY register reads 242
PHY register writes 244
PHY registers 393
PHY registers, accessing 241
PHY/link interface, packet transmission 228
Physical ID 278
physical ID 306, 356
physical ID, selection of 306
physical layer 42, 53
ping packet 206, 214, 348, 355
port disable 452
port registers - 1394a 403
port repeater 59
port status receiver 98
port status registers - 1995 397
port suspend via loss of bias 453
Port_event bit 278
power class 135
power class codes 432
power consumers 437
power consumption requirements-1995 138
power distribution 137, 431
Power Distribution specification 20
power management specification 20, 427, 429
power management, 1394-1995 428
power management, 1394a 428, 429, 459
power management, 1995 345
power management, by bus manager 345
power management, by IRM 342, 346
power management, CSR register summary 463
power management, CSR regs 461
power management, review of 1995 specification 428
power management, ROM entries 472
power providers 433
power reset 364
power state changes 274
power states 460
power states, nodes 460
power states, units 461
Power Statue Management Specification (Revision 0.71) 429
POWER_CHANGE register 468
POWER_FAIL_IMMINENT register 380
Power_Level entry 477
POWER_SOURCE register 380
primary power providers 434
priority arbitration 152, 161
PRIORITY_BUDGET register 161
Private space 28
protocol layers 17, 42, 66
read packets 175
read request - block packet 179
read request - quadlet packet 176
read response - block packet 181
read response - quadlet packet 177
read transaction 30
remote access packet 206, 215, 355
remote command packet 206, 217, 355
remote confirmation packet 206, 218, 355
remote reply packet 206, 216, 355
repeater 23, 59
Request service 45
request subaction 30
requester 30
reset duration 276
reset signaling 110, 282
reset storm 279
reset, sources of 274
RESET_START register 364, 373
RESET_TIME 283
responder 30
response code 56, 191
response packet errors 260
response packet time-out 259, 261
Response service 45
Index

response subaction 30
response transmission error 259
resume 454
resume packet 206, 219, 355, 454
resuming via port events 455
resuming via resume packet 454
resuming via resume
  port command packet 454
retry 247
retry codes 248
retry limit 249, 252
retry protocol 247
retry_A 248, 251
retry_B 256, 257, 258
retry_busy_A 252
retry_busy_B 252
retry_X 255, 257
retry_x 249
root contention 295
root dependent directories 416
root directory 416
root directory entries - required 419
Root Hold Off (RHB) 355
root hold off bit (RHB) 356, 397
root holdoff bit 213
root ID 356
root leaf entries 416
ROOT_CONTEND_FAST 295
ROOT_CONTEND_SLOW 295
RX_DISABLE_NOTIFY 452
RX_REQUEST 109, 142
RX_SUSPEND 446
S
Scalable Coherent Interface 25
SClk signal 227
self Identification (Self-ID) packet 205
self-ID packets 1-3 210, 325
self-ID count 307
self-ID packet 135, 307
self-ID packet one, two, & three 210
self-ID packet zero 207, 323
self-ID packet zero, 1394-1995 207
self-ID packet zero, 1394a 209
self-ID packets 1 & 2 211
self-ID packets, contents 323
self-ID signaling 306
self-identification overview 270
self-identification process 305
self-powered nodes 438
serial bus control services 353
short (or arbitrated) bus reset 283
short bus reset 283, 450
signal interface 102
signal velocity 89
single phase retry 249
single phase retry state machine 249, 250
single retry state transitions 249
speed exchange 312
speed map 349, 390
speed map, accessing 349
speed of transmission 17
speed signaling 117, 118, 119
SPEED_MAP 388
Split transaction 49
SPLIT_TIMEOUT register 373
STATE register 364
STATE_CLEAR register 364, 365
STATE_SET register 364, 366
storm of resets 279
Stream packet 200
strobe drivers 106
subaction 30
subaction gap 126, 142, 147, 157
suspend command packet 448
suspend initiator 446, 447, 450, 451
suspend signaling 448
suspend target 447, 451
suspend via command packet 449
suspend/resume 445
Suspend/Resume Specification
  (Revision 0.73) 429
suspended port handshake 447
suspending port via loss of bias 453
Index

suspending via port disable 451
suspending via RX_SUSPEND 451
T

topology 23
topology map 346, 389
topology map, accessing 347
TOPOLOGY_MAP registers 388
TPA 54
TPB 54
TpBias 451
transaction 30
transaction codes - asynchronous 167
transaction label 78
transaction layer 42, 45
transaction layer services 45
transaction layers 17
transaction retry 247
transaction type, packet field 56
transaction types 45
transactions 30
Transfer Speed 39
transfer types 39
tree ID process 285, 286
tree ID signaling 286
tree identification overview 268
tree-ID state 277
twisted pair signaling 54
twisted pairs 95
TX_DISABLE_NOTIFY 452
TX_GRANT 142, 144
TX_REQUEST 109, 142, 143, 275
TX_SUSPEND 446, 450
U

Unified transaction 50
unified transactions 52
unit 25
unit dependent directories 417
unit directory entries 416
unit leaf entries 417
UNIT_POWER_CONTROL Register 470
Unit_Power_Directory Entry 477
Unit_Power_Management entry 477

V

vendor identification registers 407
vendor-ID 410

W

write request - block packet 171
write request - quadlet packet 169
write response packet 173
write transaction 30