

FIREWIRE® SYSTEM ARCHITECTURE

SECOND EDITION

MINDSHARE, INC.

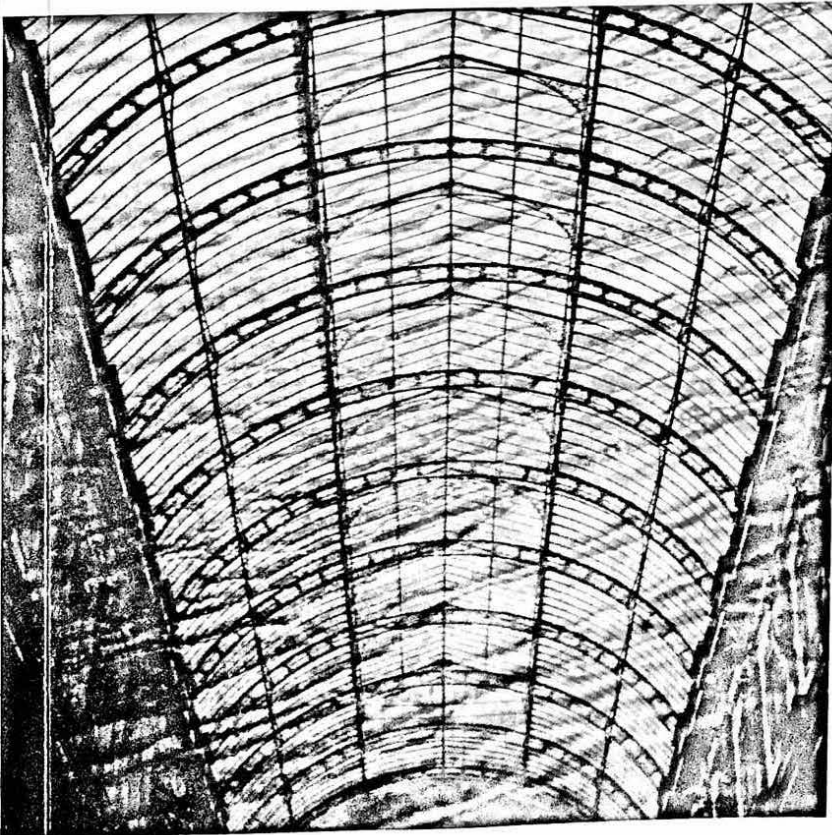


Addison
Wesley

"The 'must-have' PC architecture reference set."

—PC Magazine's "Read Only" column

FIREWIRE[®] SYSTEM ARCHITECTURE



SECOND EDITION

IEEE 1394a

MINDSHARE, INC.

Don Anderson

**PC SYSTEM
ARCHITECTURE
S E R I E S**

*FireWire®
System
Architecture,
Second Edition*

The PC System Architecture Series

MindShare, Inc.

Please see our web site (<http://www.awprofessional.com/series/mindshare>) for more information on these titles.

AGP System Architecture: Second Edition
0-201-70069-7

CardBus System Architecture
0-201-40997-6

FireWire® System Architecture: Second Edition
0-201-48535-4

InfiniBand System Architecture
0-321-11765-4

ISA System Architecture: Third Edition
0-201-40996-8

PCI System Architecture: Fourth Edition
0-201-30974-2

PCI-X System Architecture
0-201-72682-3

PCMCIA System Architecture: Second Edition
0-201-40991-7

Pentium® Pro and Pentium® II System Architecture: Second Edition
0-201-30973-4

Pentium® Processor System Architecture: Second Edition
0-201-40992-5

Plug and Play System Architecture
0-201-41013-3

Protected Mode Software Architecture
0-201-55447-X

Universal Serial Bus System Architecture: Second Edition
0-201-30975-0

HyperTransport™ System Architecture
0-321-16845-3

FireWire[®]

System Architecture,

Second Edition

IEEE 1394a
MINDSHARE, INC.

Don Anderson



ADDISON-WESLEY

Boston • San Francisco • New York • Toronto • Montreal
London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The authors and publisher have taken care in the preparation of this book but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers discounts on this book when ordered in quantity for bulk purchases or special sales. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the U.S., please contact:

International Sales
international@pearsoned.com

Visit us on the Web: www.awprofessional.com

Library of Congress Cataloging-in-Publication Data

Anderson, Don, 1953-..

FireWire systems architecture: IEEE 1394a / Mindshare, Inc.;
Don Anderson. —2nd ed.

p. cm.

Includes bibliographical references and index.

ISBN 0-201-48535-4

1. IEEE 1394 (Standard) I. Mindshare, Inc. II. Title.

TK7895.B87 A52 1998

621.39'81—dc21

98-43465

ISBN: 0-201-48535-4

Copyright © 1999 by Mindshare, Inc.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

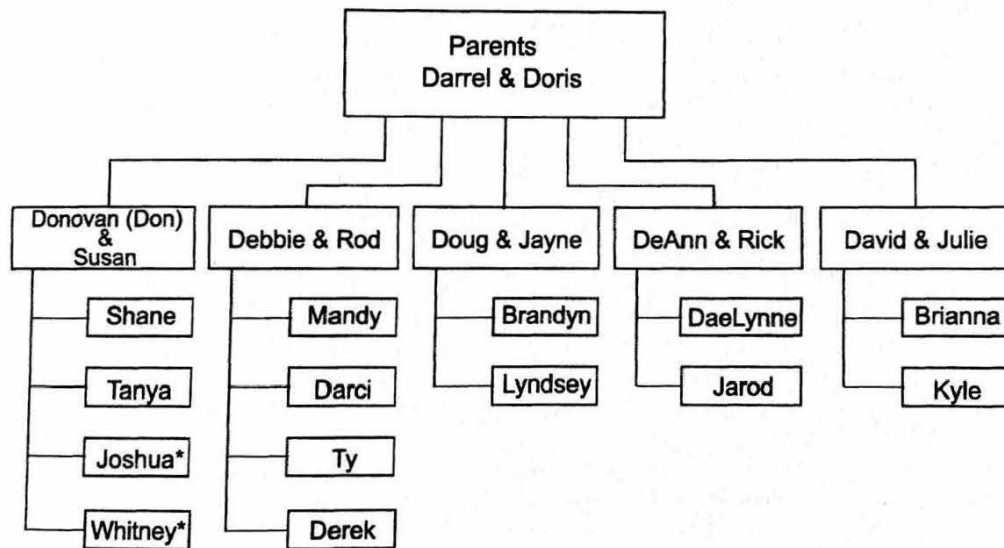
Pearson Education, Inc.
Rights and Contracts Department
One Lake Street
Upper Saddle River, NJ 07458

ISBN 0201485354

Text printed in the United States on recycled paper

Eighth printing, May 2005 at Offset Paperback Manufacturers in Laflin, Pennsylvania

For my sisters Debbie and DeAnn and my brothers Doug and David, and their families.



* Step children

The PC System Architecture Series

The PC System Architecture Series is a crisply written and comprehensive set of guides to the most important PC hardware standards. Each title illustrates the relationship between the software and hardware, and thoroughly explains the architecture, features, and operation of systems built using one particular type of chip or hardware specification.

MindShare, Inc. is one of the leading technical training companies in the computer industry,

providing innovative courses for dozens of companies, including Intel, IBM, and Compaq.

"There is only one way to describe the series of PC hardware and architecture books written by Tom Shanley and Don Anderson: INVALUABLE."
—PC Magazine's "Read Only" column



ISBN 0-201-40994-1



ISBN 0-201-70069-7



ISBN 0-201-40997-6



ISBN 0-201-40995-X



ISBN 0-201-48535-4



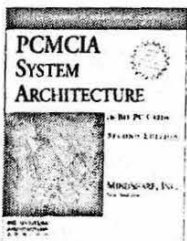
ISBN 0-201-40996-8



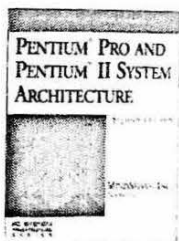
ISBN 0-201-30974-2



ISBN 0-201-72682-3



ISBN 0-201-40991-7



ISBN 0-201-30973-4



ISBN 0-201-40992-5



ISBN 0-201-41013-3



ISBN 0-201-40990-9



ISBN 0-201-55447-X



ISBN 0-201-30975-0

<http://www.awl.com/cseng/series/mindshare/>

Addison-Wesley

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	24
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.....	77
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)	250
Receiving-Node Retry Behavior (Inbound Retry)	251
Dual Phase Retry.....	252
Sending-Node Retry Behavior (Outbound Retry)	255
Receiving-Node Retry Behavior (Inbound Retry)	259
Transactions Errors	259
Packet Transmission Errors	262
Packet Error Handling Summary	

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	277
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

Contents

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

Contents

Bus_Manager_ID Register	383
Bandwidth_Available Register	384
Channels_Available Register	384
Maint_Control Register	386
Maint_Utility Register	386
Unit Registers.....	388
Topology Map	389
Speed Map.....	390

Chapter 22: PHY Registers

Overview.....	393
1394-1995 PHY Register Map	394
Port Status Registers	397
PHY Configuration Packet	397
Root Hold Off	397
Gap Count Optimization	397
1394a PHY Register Map.....	398
Page Select.....	402
Port Status Register Page.....	403
Vendor Identification Register Page	407
Vendor-dependent Page	408

Chapter 23: Configuration ROM

Overview.....	409
Minimal ROM Format.....	410
General ROM Format.....	410
Header Information	411
Info_Length.....	411
CRC_Length	412
CRC_Value.....	412
Bus_Info_Block (1394-1995)	412
Bus_Name Field.....	413
Bus Characteristics Fields	413
Node_Vendor_ID Field	414
Chip_ID Fields	415
Bus Info Block (1394a)	415
Power Management Capable	415
Generation Field.....	416
Link Speed	416
Root_Directory	416
Required Root Directory Entries	419

Contents

Module_Vendor_Id	419
Node_Capabilities	419
Node_Unique_Id	420
Unit_Directory & Unit_Power_Requirements	422
Optional Root Directory Entries	422
Bus_Dependent_Info	422
Module_Hw_Version	422
Module_Spec_Id	422
Module_Sw_Version	422
Company ID Value Administration	423

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

Contents

Unit Power Level Entry	477
Unit Power Management Entry	477
Battery Group Entry (Unit)	478
Appendix: Example 1394 Chip Solutions	
Overview	479
1394 in the PC	479
TSB12LV22 / OHCI-Lynx	480
Features	480
Overview	481
Block Diagram	482
TSB41LV03	482
Features	483
TSB41LV03 Overview	483
Block Diagram	487
Putting it all Together	488
1394 in the Digital Camera	489
TSB12LV31 – GPLynx	489
Features	489
Overview	490
Block Diagram	491
TSB21LV03A	492
Features	492
Block Diagram	493
Putting it all Together	494
For More Information	494
Appendix: Glossary	495
Index	503

Figures

2-1	PC with IEEE 1394 Bus Attached to the PCI Bus	22
2-2	IEEE 1394 Nodes May Extend the Bus Via Additional Ports	23
2-3	Module, Node, and Unit Architecture	26
2-4	CSR Address Space with IEEE 1394 Extensions	27
2-5	Serial Bus Address Space	29
2-6	Transaction Model Consisting of Request and Response Subactions	31
2-7	Isochronous Transactions Consist Only of a Request Transaction	32
3-1	Request/Response Protocol	38
3-2	Isochronous Transaction that Consists Only of a Request Transaction	39
3-3	Isochronous and Asynchronous Transactions Share Bus Bandwidth	40
3-4	Protocol Layers	43
3-5	Transaction Layer Communication	46
3-6	Acknowledge Packet is Added by the Transaction Layer	47
3-7	Link Layer Provides an Interface Between the Transaction and Physical Layers	48
3-8	Request and Response with Acknowledge Packet Included	49
3-9	Link Layer Actions Taken During Split Write Transaction	50
3-10	Conceptual Illustration of a Concatenated Transaction	51
3-11	Unified Transaction Format	53
3-12	Example of PHY Chip in a Node	54
3-13	Nodes Coordinate Use of the Serial Bus During Various Phases of Operation	55
3-14	Stages Involved in Configuration Process	55
3-15	Packets Used During Asynchronous Split Write Transactions	57
3-16	Packets Used During Asynchronous Read Transactions	57
3-17	Packets Used During Asynchronous Lock Transactions	58
3-18	Packet Used During Isochronous Transactions	58
3-19	Multiport Nodes Have Repeaters Used to Retransmit Cable Traffic	60
3-20	Example Asynchronous Read Transaction	62
3-21	Example Isochronous Transaction	63
4-1	Example Asynchronous Read Transaction	68
5-1	6-Pin Plug and Socket	86
5-2	4-Pin Plug and Socket	88
5-3	Cross-section of 6-Conductor Cable	89
5-4	6-Pin Connector Cable/Plug Assembly	90
5-5	Cross-section of 4-Conductor Cable	91
5-6	Cable/Plug Assembly with Identical 4-Pin Plugs	91
5-7	Cable/Plug Assembly with a 4-Pin Plug and 6-Pin Plug	92
6-1	Bias Voltage Applied to the Cable Permits Detection of Node Attachment/Detachment	98
6-2	Port Connection Detect Circuitry	100
6-3	The IEEE 1394-1995 Signal Interface	102
6-4	1394a Port Interface	103
6-5	Arbitration Signaling	106
6-6	Arbitration Comparator Outputs Must Be Decoded by Associated Arbitration Logic	108
6-7	Child Node Sending Transmit Request to its Parent Node	113
6-8	Parent Node Sending Transmit Grant While Child Continues Driving Transmit Request	114

Figures

6-9	Packet Framed By Data_Prefix and Data_End	115
6-10	Packet Framed By Two Data_Prefixes During Concatenated Packet Transmission	115
6-11	Topology with Speed Information Included	119
6-12	Speed Signaling Circuitry for S400 PHY	120
6-13	Timing Relationship Between Speed Signaling and Data Prefix Signaling	121
6-14	Data-Strobe Signaling During Packet Transmission	123
6-15	NRZ Data Stream	124
6-16	Data-Strobe Encoding/Decoding	125
6-17	Split Asynchronous Transaction with Gaps Defined	126
6-18	Isochronous Transaction with Gaps Defined	127
6-19	Cable Power Class Reported Via Self-ID Packet	135
6-20	Serial Bus Power Distribution	138
7-1	Two Nodes Start Arbitration by Signaling an Arbitration Request	144
7-2	Arbitration Signaling Reaches Root Node and TX_GRANT Returned	145
7-3	End of Arbitration / Beginning of Packet Delivery	146
7-4	Asynchronous Arbitration and the Fairness Interval	149
7-5	Example of Isochronous Transaction Arbitration	151
7-6	Example of Fairness Interval Combined with Isochronous Transactions	153
7-7	Cycle Start Skew	155
7-8	Cycle Start Packet Jitter	156
7-9	Example of Acknowledge Accelerated Arbitration	158
7-10	Fly-By Arbitration Performed by Three-Port Node	160
7-11	FAIRNESS_BUDGET Register Format	162
8-1	Primary Asynchronous Packet Format	166
8-2	Write Request — Quadlet Format	169
8-3	Write Request — Block Format	171
8-4	Write Response Packet Format	173
8-5	Format of Asynchronous Stream Packet	175
8-6	Read Request — Quadlet Packet Format	176
8-7	Read Response — Quadlet Packet Format	177
8-8	Read Request — Block Packet Format	179
8-9	Read Response — Block Packet Format	181
8-10	Lock Request Packets	184
8-11	Lock Response Packet Format	189
8-12	Acknowledge Packet	191
8-13	Split Write Transaction	193
8-14	Concatenated Split Write Transaction	193
8-15	Unified Write Transaction	194
8-16	Normal Read or Lock Transaction	194
8-17	Concatenated Read or Lock Transaction	195
8-18	Cycle Start Packet Format	196
9-1	Format of an Isochronous Stream Packet	200
9-2	Non-Concatenated Isochronous Transactions	202
9-3	Concatenated Isochronous Transactions - If Sent by Same Node	203
10-1	General Format of PHY Packet	206
10-2	Format of Self-ID Packet Zero	208

Figures

10-3	Format of Self-ID Packets One-Three.....	211
10-4	Format of Self-ID Packets One and Two — 1394a	212
10-5	Link-on Packet Format.....	212
10-6	PHY Configuration Packet Format.....	213
10-7	Ping Packet Format.....	215
10-8	Remote Access Packet Format	216
10-9	Remote Reply Packet Format.....	217
10-10	Remote Command Packet Format.....	217
10-11	Remote Confirmation Packet Format.....	218
10-12	Resume Packet Format.....	220
11-1	Interface Between the Link and PHY.....	222
11-2	The Link Request Specifies the Speed of the Upcoming Data Transmission.....	230
11-3	LReq & Ctl Timing Relationship.....	233
11-4	Sequence and Format of Data Lines During Packet Receive State	237
11-5	Format and Content of Data & Control Signals During Status Transfers	238
11-6	1394a PHY Register Definition.....	242
11-7	Register Contents Returned During Status Transfer	244
12-1	BUSY_TIMEOUT Register Showing Single Phase Retry Count	249
12-2	Outbound Retry State Machine — Single Phase	250
12-3	Inbound Retry State Machine — Single Phase	251
12-4	BUSY_TIMEOUT Register Showing Dual Phase Time-out Value.....	252
12-5	Outbound Retry State Machine — Dual Phase	255
12-6	Inbound Retry State Machine — Dual Phase.....	258
12-7	Split Transaction and Related Packets.....	259
13-1	Overall Cable Configuration Time.....	266
13-2	Example of Bus Topology after Bus Reset.....	268
13-3	Example of Bus after New Device Is Attached and Bus Is Reset.....	269
13-4	Example Bus Following Tree Identification.....	270
13-5	Example Bus Following Self-ID Process.....	271
14-1	Reset Signaling and Detection	276
14-2	Nodes After Reset Have No Sense of Bus Topology	277
14-3	Timing Relationships Between 1394-1995 and 1394a Reset Detection & Signaling.....	281
14-4	Arbitrated Reset Timing	284
15-1	Strobe and Data Lines Used During Tree ID	287
15-2	Leaf Node Signaling Parent_Notify	288
15-3	Branch Node Signaling Child_Notify.....	289
15-4	Example Node Network Following Bus Initialization.....	290
15-5	Leaf Nodes Signal Parent_Notify	291
15-6	Node Network Following Leaf Node's A & E Having Identified Parents.....	291
15-7	Branch Nodes Attempt to Locate Their Parents.....	292
15-8	Final Topology — Scenario 1	293
15-9	Example Node Network Following Reset — Scenario Two.....	294
15-10	All Leaf Nodes Identified as Children.....	294
15-11	Root Contention — Two Attached Nodes Signaling Parent_Notify to the Other	296
15-12	Example Topology after Completing Tree ID Process	297
15-13	Format of PHY Configuration Packet.....	298

Figures

15-14	Family of Nodes after Reset with Node E's Force Root Bit Set	299
15-15	Topology Following Leaf Identification	300
15-16	Topology Following Completion of Stage One of the Branch Identification Process	301
15-17	Topology Following Stage Two of the Branch Identification Process	302
15-18	Node with the Force Root Bit Set Becomes the Root Node	303
15-19	Example of Looped Topology Resulting in a DeadLock During Tree ID	304
16-1	Example Node Network at Start of Self-ID	307
16-2	First Arbitration Grant Issued During Self-ID Process	308
16-3	Identified Node Starts Self-ID Packet Transmission with Data Prefix	309
16-4	Self-ID Packet Is Broadcast to All Nodes.....	310
16-5	Node A Signals Identification Done to End Its Self-Identification Process.....	311
16-6	State of the Node Network Following the First Physical ID Assignment.....	313
16-7	State of the Node Network Following the Second Physical ID Assignment	315
16-8	State of the Node Network Following the Third Physical ID Assignment	317
16-9	State of the Node Network Following the Fourth Physical ID Assignment	319
16-10	State of the Node Network Following the Fifth Physical ID Assignment.....	321
16-11	State of the Node Network Following the Final Physical ID Assignment	322
16-12	Format for Self-ID Packet Zero	323
16-13	Format of Self-ID Packets One and Two	325
16-14	Protocol Used to Transmit Self-ID Packets	326
17-1	Isochronous Transactions Performed Every 125 μ s.....	330
17-2	Cycle Start Packet Contains Value of Cycle Master's CYCLE_TIME Register	331
17-3	Cycle Time Variation Included in Cycle Start Packet.....	331
18-1	Contender Nodes Must Set Bits l and c in Their Self-ID Packets.....	334
18-2	Location of CHANNEL_ALLOCATION & BUS_BANDWIDTH Registers	336
18-3	Format of the CHANNELS_AVAILABLE Register	337
18-4	Operation of Lock Compare & Swap Transaction during Channel Allocation.....	339
18-5	Format of the BANDWIDTH_AVAILABLE Register.....	340
18-6	Operation of Lock Compare & Swap Transaction during Isochronous Bandwidth Allocation.....	341
19-1	Format of the Link-On Packet Used to Apply Bus Power to a Node's Link Layer.....	346
19-2	Topology Map Format	347
19-3	Format of PHY Configuration Packet	348
19-4	Ping Packet Format.....	349
19-5	Speed Map Format.....	350
20-1	Relationship Between Bus Management Layer and the Rest of the Node	352
21-1	Location of CSRs Within the Node's Address Space.....	362
21-2	Format of the STATE Register.....	365
21-3	Format of the Bus_Dependent Field within the STATE Register	367
21-4	NODE_IDS Register Format.....	370
21-5	Bus_id and Offset_id Fields Define the Address Space Allocated to a Given Serial Bus and Node	372
21-6	Format of RESET_START Register	373
21-7	Format of the SPLIT_TIMEOUT Register.....	374
21-8	Format and Relationship Between CYCLE_TIME & BUS_TIME Registers	378
21-9	Contents of Cycle Start Packet	379

Figures

21-10	Actions and Their Effects on the CYCLE_TIME Register	379
21-11	Actions and Their Effects on the BUS_TIME Register	380
21-12	Format and Behavior Summary of the POWER_FAIL_IMMINENT Register	381
21-13	Format and Behavior Summary of the POWER_SOURCE Register	381
21-14	Format and Behavior Summary of BUSY_TIMEOUT Register	382
21-15	Format and Behavior of the BUS_MANAGER_ID Register	383
21-16	Format and Behavior Summary of the BANDWIDTH_AVAILABLE Register	384
21-17	Format and Behavior Summary of CHANNELS_AVAILABLE Register	385
21-18	Format and Behavior of MAINT_CONTROL Register	387
21-19	Format and Behavior of MAINT_UTILITY Register	387
21-20	Format of TOPOLOGY_MAP	390
21-21	Format of SPEED_MAP	391
22-1	1394-1995 PHY Register Format	394
22-2	Format of PHY Configuration Packet	398
22-3	1394a PHY Register Definition	399
22-4	Format of Port Registers	403
22-5	Port Disable Feature and Its Effect	406
22-6	Format of the Vendor Identification Page	407
23-1	Minimal ROM Format	410
23-2	General ROM Format	411
23-3	Format of the Bus_Info_Block	412
23-4	Illustration of the Bus Information Block for 1394a Compliant Nodes	415
23-5	Example Root Directory Tree	417
23-6	Root Directory Entries	418
23-7	Format of Node_Capabilities Root Directory Entry	419
23-8	Format and Contents of the Node_Unique_Id Entry and Leaf	421
23-9	CSR & Serial Bus Unique Identifiers	422
25-1	Power Providers Must Place Diodes in Each Port VG Line	434
25-2	Configuration of Power Provider	435
25-3	Alternate Power Provider with Bus Powered PHY (Power Class 4)	437
25-4	Power Consumer Node Implementation	438
25-5	Example of Self-Powered Node (Class 0)	439
25-6	Example of Two-Port Node that Does Not Use Cable Power (Class 4)	440
25-7	Example of Node with 3 Ports that Does Not Use Cable Power (Class 4)	441
25-8	Example of Self-Powered Node that Uses Cable Power if Main Power is Lost	442
25-9	Example Self-Powered Node with PHY Powered by Cable	443
26-1	PCI to 1394 Bridge Node Sends Suspend Command to VCR Node, Port 2	446
26-2	Actions Taken by the Suspend Initiator	447
26-3	State of Network Following Bus Reset	448
26-4	"Initiate Suspend" Command Packet Format	449
26-5	Suspend Confirmation Packet Format	449
26-6	Action Taken by Suspend Initiator and Suspend Target Following Confirmation Packet	450
26-7	Format of "Disable Port" Command Packet	452
26-8	Illustration of Actions Taken By Target Node When Port Disable Command is Issued	453
26-9	Format of a Resume Packet	454

Figures

26-10	Format of a Command Packet Defined as a Resume Port Packet	455
26-11	Port Registers and Port Event Enable Bits.....	456
26-12	PHY Registers.....	456
27-1	Format of the NODE_POWER_CONTROL Register	457
27-2	Format of the Notification Address Register	465
27-3	Format of the Cable_Power_Source_State Register	466
27-4	Format of the CABLE_POWER_SOURCE_CONTROL Register	467
27-5	Format of the POWER_CHANGE Register	467
27-6	Format of UNIT_POWER_STATE Register	469
27-7	Format of UNIT_POWER_CONTROL Register	470
27-8	Format of the BATTERY_STATE Register	470
27-9	ROM Power Directories	472
27-10	Format of the Node_Power_Directory Entry.....	473
27-11	Format of the Node_Power_Level Entry	474
27-12	Format of the Cable_Power_Source_Level Entry	474
27-13	Format of the Node_Power_Management Entry	475
27-14	Format of Battery_Group Entry	476
27-15	Format of the Battery_State Entry	476
27-16	Format of the Unit_Power_Directory Entry within the Unit Directory.....	477
27-17	Format of the Unit_Power_Level Entry.....	477
27-18	Format of the Unit_Power_Management Entry	478

Tables

1-1	FireWire Key Features	17
3-1	Maximum Data Block Size for Asynchronous Transfers.....	41
3-2	Maximum Data Block Size for Isochronous Transfers	42
4-1	Service Used During Asynchronous Request Subactions	66
4-2	Service Used During Response Subaction	73
4-3	Services Available for Isochronous Applications	80
5-1	Contact Signal Assignments and Numbers	87
6-1	Port Status Voltage States.....	99
6-2	Differential Output Signal Amplitude.....	104
6-3	Differential Receive Signal Amplitude.....	105
6-4	Arbitration Signal Generation Rules.....	107
6-5	Arbitration Comparator Signal Decoding Rules.....	109
6-6	Bus Reset Signaling.....	110
6-7	Signaling States Used During Tree Identification.....	111
6-8	Signaling States Used During Self-Identification.....	111
6-9	Arbitration Line States During Normal Arbitration.....	112
6-10	Data Prefix and Data End Signaling	115
6-11	Port Disable and Suspend Line State Signaling	117
6-12	Common Mode Current for Speed Signaling.....	122
6-13	Gap Timing	127
6-14	1394-1995 Cable Interface Timing Constants	128
6-15	1394a Cable Interface Timing Constants	130
6-16	Cable Power Requirements — 1394-1995.....	134
6-17	Cable Power Requirements — 1394a.....	134
6-18	Contents of the Power Field within the Self-ID Packet (1394-1995).....	136
6-19	Definition of Power Class Values Within “Pwr” Field of Self-ID Packet—1394a	136
7-1	Arbitration Line States During Normal Arbitration.....	143
7-2	Definition of FAIRNESS_BUDGET Register Fields.....	162
8-1	Asynchronous Transaction Codes.....	167
8-2	Maximum Data Payload for Asynchronous Packets.....	168
8-3	Write Request — Quadlet Packet Components	169
8-4	Write Request — Block Packet Components	172
8-5	Write Response Packet Components	173
8-6	Read Request — Quadlet Packet Components	176
8-7	Read Response — Quadlet Packet Components.....	178
8-8	Read Request — Block Packet Components	179
8-9	Read Response — Block Packet Components	182
8-10	Lock Request Packet Components	185
8-11	Extended Transaction Code Field Definition for Lock Transactions	186
8-12	Data Length Field in Lock Request Packets.....	187
8-13	Lock Response Packet Components	190

Tables

8-14	Definition of Response Code Values within Asynchronous Packets.....	191
8-15	Acknowledge Code Values	192
8-16	Cycle Start Packet Contents	196
9-1	Isochronous Stream Packet Components.....	201
9-2	Maximum Data Payload for Isochronous Packets.....	202
10-1	Contents of the Self-ID Packet Zero — 1394-1995	208
10-2	Definition of Power Class Values Within "Pwr" Field of Self-ID Packet—1394a	209
10-3	Definition of Self-ID Packets One - Three (1394-1995)	211
10-4	Remote Access Packet Field Description Summary	216
10-5	Remote Command Packet Field Definitions.....	218
10-6	Remote Confirmation Packet Field Description.....	219
11-1	Link/PHY Signal Interface.....	223
11-2	Control Signal Definition with PHY Driving	224
11-3	Control Signal Definition with Link Driving.....	227
11-4	Bus Request Format for Cable Environment - 1394-1995 Standard	229
11-5	Bus Request Format for Cable Environment - 1394a Standard.....	230
11-6	Speed Signaling Link to PHY.....	231
11-7	Rules Governing When Link May Initiate a Request on LReq	233
11-8	Link Requests and Corresponding PHY Behavior and Link Actions	235
11-9	Speed Signaling PHY to Link	237
11-10	Status Bits Returned Via D[0:1]	239
11-11	Acceleration Control Request Format	241
11-12	Register Read Request Format.....	243
11-13	Register Write Request Format.....	244
12-1	Retry Codes Returned by Busy Node.....	248
15-1	Line States that Signal Parent- and Child-Notify	286
16-1	Line States that Signal Parent- and Child-Notify	306
16-2	Possible Speed Signaling Combinations	312
16-3	Self-ID Packet Zero Field Definition	323
21-1	Core CSR Locations and Definition	363
21-2	Field Definitions for Register	365
21-3	Definition of the Bits within the Bus_Depend Field of the STATE Register.....	368
21-4	Definition of the Fields within the NODE_IDS Register	370
21-5	Serial Bus Dependent CSR Register Locations and Definition	376
21-6	Addresses Allocated for the TOPOLOGY_MAP and SPEED_MAP.....	388
22-1	Description of the PHY Register Map For 1394-1995	395
22-2	Description of the PHY Register Map For 1394a.....	399
22-3	Description of the Port Status Page.....	404
22-4	Description of the Vendor Identification Page.....	408
23-1	max_rec values and Maximum Data Payload Sizes	414
23-2	Definition of key_type and entry_values	418

Tables

23-3	Definition of Bit Fields within the Node_Capabilities Entry	419
25-1	Node Power Class, Code Assignments For Each Power Configuration	432
25-2	Definition of Power Class Values Within "Pwr" Field of Self-ID Packets	432
25-3	Cable Power Requirements	433
27-1	Node Power States	460
27-2	Unit Power States	462
27-3	Node-Specific Power-Related CSRs	463
27-4	Unit-Specific Power-Related CSRs	464
27-5	Definition of Function Field Within the NODE_POWER_CONTROL Register	465
27-6	Definition of Function Field Within the NODE_POWER_CONTROL Register	468
27-7	Definition of Function Field Within the UNIT_POWER_CONTROL Register	471

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.

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

Scalable Performance	Speeds of 100, 200, and 400 Mb/s supported.
Hot Insertion & Removal	Devices can be attached or removed from the bus dynamically without powering the system down.
Plug and Play	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).
Support for two types of transactions	Support for isochronous and asynchronous transfers.
Layered hardware and software model	Communications based on a transaction layer, link layer, and physical layer protocols.
Support for 64 nodes	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.
Address space of 16 petabytes per bus	Each of the 64 nodes has 256TB of address space, making the total address space of 16 petabytes.
Support for 1024 buses	The CSR architecture supports up to 1024 buses for a total address space of 16 exabytes.
Peer-to-Peer transfer support	Serial bus devices have the ability to perform transactions between themselves, without the intervention of a host CPU.

FireWire System Architecture

Table 1-1: FireWire Key Features (Continued)

Supports fair arbitration	Implements arbitration to ensure that isochronous applications are guaranteed a constant bus bandwidth, while asynchronous applications are permitted access to the bus based on a fairness algorithm.
Error detection and handling	CRC are performed to verify successful transmission of data across the serial bus. CRC errors are detected and transactions retried where possible.
Employs two twisted pairs for signaling.	Signaling is performed using two twisted pairs: one pair for data transmission and another for synchronization.
Cable power	Power available from the bus can be either sourced or sunk by a given node.
Expandable bus	The serial bus can be extended by connecting new serial devices to ports provided by serial bus nodes. Nodes that have two or more ports are termed branch nodes, which can daisy chain additional nodes to the bus. Nodes implementing a single port are termed leaf nodes, which represent the termination point of a given branch of the serial bus.

2

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:

- CSR Architecture Specification — ISO/IEC 13213 (ANSI/IEEE 1212)
- IEEE 1394-1995 Serial Bus Specification
 -  • 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-

tion has been developed, called 1394a. This supplement also adds additional features and makes improvements intended to increase performance or usability. This text incorporates the changes defined by 1394a. However, at the time of this writing 1394a was not an officially released document from IEEE.

IEEE 1394.B

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

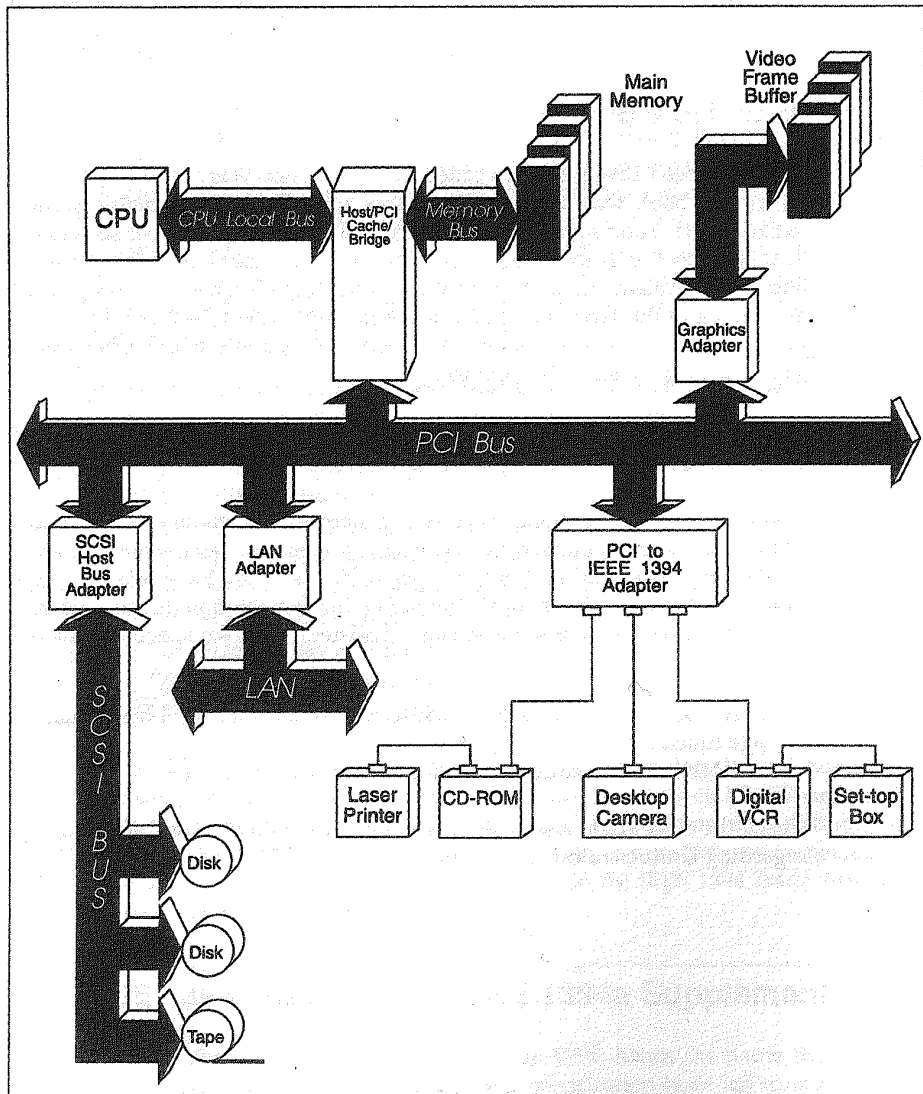
A wide variety of functional devices (e.g. hard drives, video cameras, and CD-ROMs) can be implemented as 1394 nodes. Functional devices are termed units by the 1394 specification. Certain types of devices may have related specifications called “unit architectures” that define implementation details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

FireWire System Architecture

Figure 2-1: PC with IEEE 1394 Bus Attached to the PCI Bus.



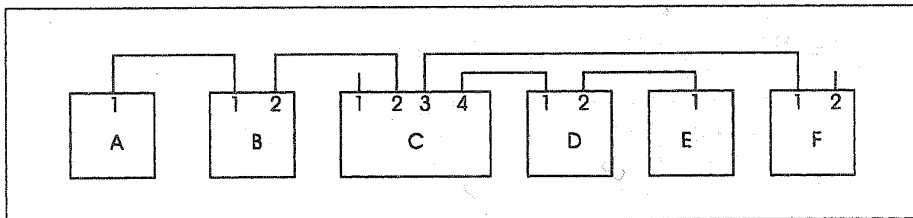
IEEE 1394 Topology

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.

Figure 2-2: IEEE 1394 Nodes May Extend the Bus Via Additional 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.

FireWire System Architecture

Peer-To-Peer Transfers

Unlike most other serial buses designed to support peripheral devices (e.g. Universal Serial Bus) peer-to-peer transfers are supported so that serial bus nodes can transfer data between each other without intervention from a host system. This enables high throughput between devices without adversely affecting performance of the computer system. For example, a video camera can set up a transfer between itself and a video cassette recorder, when both reside on the same serial bus or bridged serial buses.

Device Bay

Device bay provides a standard mechanism for attaching serial bus devices into desktop systems. The device bay is designed specifically for PCs and allows attachment of serial bus devices (including Firewire and Universal Serial Bus (USB) devices) into a docking bay integrated into the computer system, rather than requiring a cable attachment. This is intended for devices such as hard drives, DVD drives, or modems that can be added to a desktop system without having to power the system down and load drivers. Specifications for Device Bay implementations can be found at: www.device-bay.org.

The ISO/IEC 13213 Specification

The ISO/IEC 13213 (ANSI/IEEE 1212) 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, including IEEE 1394. The primary goals of the ISO/IEC 13213 specification are to:

- Reduce the amount of customized software needed to support a given bus standard.
- Simplify and improve interoperability of bus nodes based on different platforms.
- Support bridging between different bus types.
- Improve software transparency between multi-bus implementations.

Working groups involved in the development of IEEE 1394 (FireWire), IEEE 896 (Futurebus+), and IEEE 1596 (Scalable Coherent Interface) have helped define and have adopted the ISO/IEC 13213 specification. The specification defines the following features:

- Node Architectures
- Address space
- Common transaction types
- Control and Status Registers (CSRs)
- Configuration ROM format and content
- Message broadcast mechanism to all nodes or to units within a node
- Interrupt broadcast to all nodes.

The ISO/IEC 13213 specification also permits bus-dependent extensions and features that are defined by the IEEE 1394 serial bus specification. The following sections discuss the specific ISO/IEC 13213 features implemented by the IEEE 1394 specification.

ISO/IEC 13213 and IEEE 1394 are based on big endian memory addressing convention. Consequently, this book also follows the big endian convention to minimize the difficulty in translating between this book and the specifications.

Note: To simplify terminology the ISO/IEC 13213 specification is referred to as the “CSR architecture” in this text.

Node Architecture

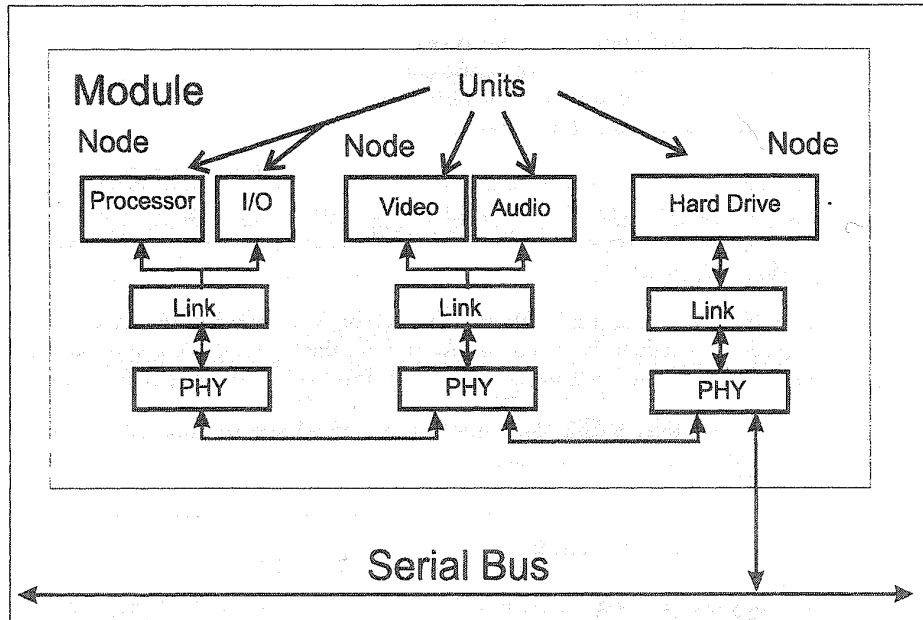
The physical and logical organization of devices attached to a compliant bus is represented by particular terminology:

- Module — represents a physical device attached to the bus, which contains one or more nodes.
- Node — represents a logical entity within a module. Nodes are visible to the initialization software and contain CSRs and ROM entries that are mapped into the initial node address space. After the system has been initialized, most of the node’s CSRs and ROM entries are no longer needed. Some of the node registers are shared between units within the node.
- Unit — represents the functional subcomponents of a node that may identify either processing, memory, or I/O functionality. Units within a node typically operate independently and are controlled by their own software drivers. Registers defined by a given unit are mapped into the node address space and are accessed by a unit specific software driver.

FireWire System Architecture

Figure 2-3 on page 26 illustrates the node architecture and reflects the units that can be implemented within a node. The number of units within a node is design dependent. As an example, a module might be a video tape deck with separate nodes for audio, video, and controls. Each node might define a unit architecture that specifies high level protocols needed for transferring data.

Figure 2-3: Module, Node, and Unit Architecture



FireWire System Architecture

Transfers and Transactions

The nature of a given node application dictates the type of transfer(s) that it performs. Some applications may require periodic data transfer with guaranteed data delivery (asynchronous), while the rate of transfer may be crucial to other applications (isochronous). The serial bus supports both isochronous and asynchronous data transfer protocols.

The serial bus must share bus bandwidth among a variety of nodes residing on the bus. To accomplish bus bandwidth sharing, a given transfer is typically performed as a series of smaller transactions that occur over time. Isochronous transfers require transactions that occur at a constant rate, while asynchronous transactions may occur periodically.

Asynchronous Transfers

Some serial bus applications require data transfers to occur without any data corruption. Data transfers must be valid, otherwise the integrity of the application is compromised, or worse the entire system may fail. The serial bus provides verification of valid data delivery across the serial bus, as well as confirmation back to the initiator that the transfer was successfully received by the application. In the event that data is not received correctly, the failure is reported to the initiator of the transfer, which can retry the transfer.

The CSR architecture defines three basic transaction types that are used by the serial bus for asynchronous transfers:

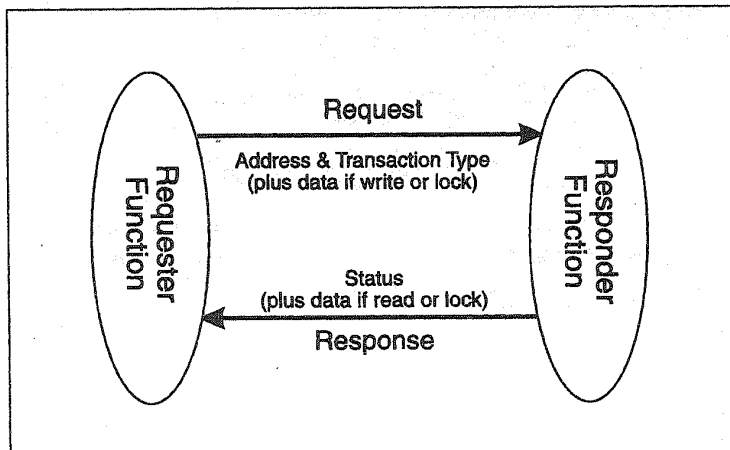
- Reads
- Writes
- Locks

Due to the need to confirm data delivery, asynchronous transactions require a more complex bus protocol. An asynchronous bus transaction is initiated by a **requester** node and received by a **responder** node that returns a response. Thus, each **transaction** consists of two **subactions**:

1. request subaction — transfers the address, command, and data (writes and locks) from the requester to the responder.
2. response subaction — returns completion status (writes) back to the requester or returns data during read and lock transactions.

Figure 2-6 illustrates the transaction model. Note that the IEEE 1394 specification adds additional protocol layers that require read transactions to be split into two separate operations. Writes can be performed either as unified or split operations depending on the speed of the protocol layers. Various forms of transactions are discussed in the next chapter.

Figure 2-6: Transaction Model Consisting of Request and Response Subactions



The locked transaction is a mechanism that permits read-modify-write operations without the implementation of a typical bus lock signal. Rather, the requester uses the locked transaction to notify the responder of its desire to perform an atomic operation. The responder is responsible for performing the test and set operation. See Chapter 8 for details regarding the implementation of locked operations.

The specification also defines error codes that are returned by the responder to the requester. Four standard code types are defined:

- `type_error`
- `address_error`
- `conflict_error`
- `response_timeout`

The error checking mechanism and the details regarding the implementation of these status codes are not defined by the specification. Details regarding the IEEE 1394 implementation of error handling are discussed in Chapter 12.

FireWire System Architecture

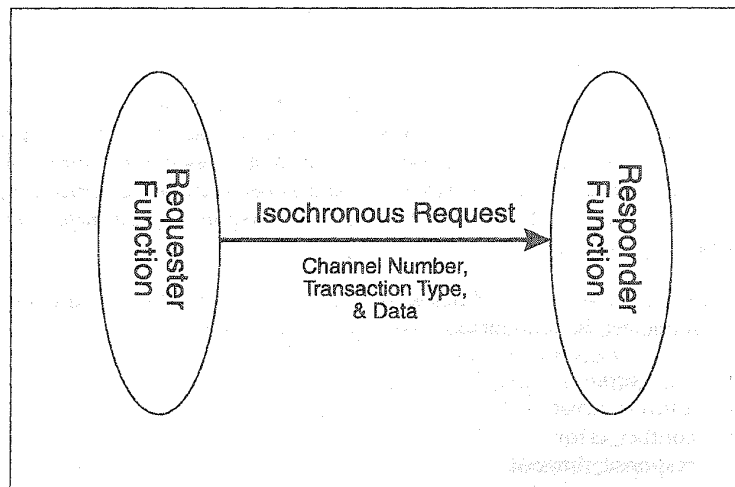
Isochronous Transfers

Isochronous applications, requiring that data be delivered at a constant rate across the bus, do not require confirmation of data delivery. For example, audio data being transferred from a music CD to a speaker via the serial bus must occur at a constant rate in order to reproduce the sound at the speaker without distortion. The audio data has a short life, therefore, if audio data is corrupted when transferred over the bus, there is no long-lasting effect. Furthermore, such applications do not need confirmation that the data has been received correctly.

Due to the nature of isochronous applications, the associated bus transactions are quite simple. An isochronous transaction sends data to a target device at regular intervals (every 125 μ s) and receives no feedback of any type from the receiving node. Consequently, a single isochronous transaction type is defined.

Figure 2-7 on page 32 illustrates the communications model used for isochronous transfers. Note that no response is returned. Note that the requester function is known as an isochronous talker and the responder function is called an isochronous listener.

Figure 2-7: Isochronous Transactions Consist Only of a Request Transaction



3

Communications Model

The Previous Chapter

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

This Chapter

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

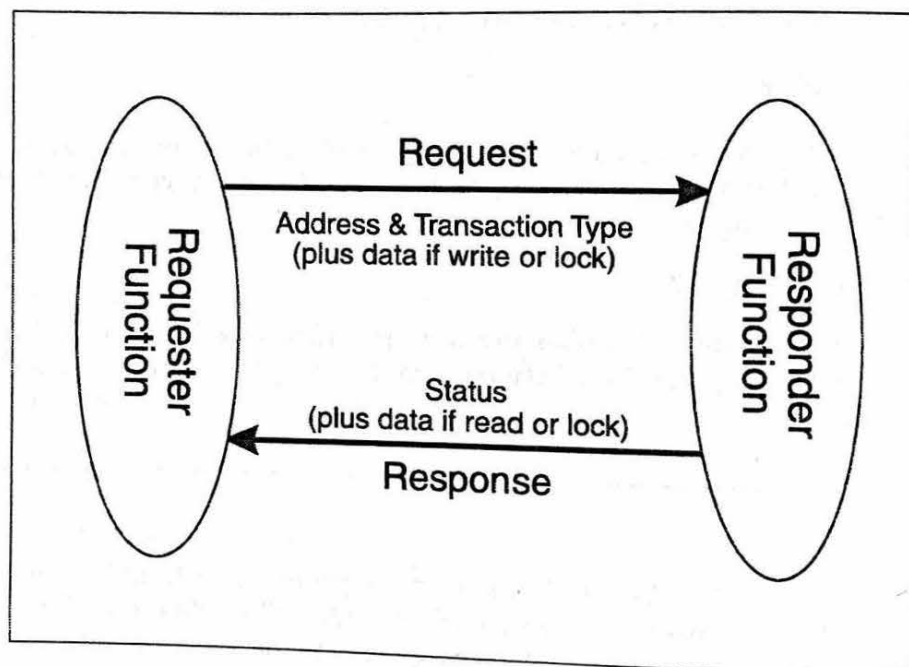
FireWire System Architecture

- 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

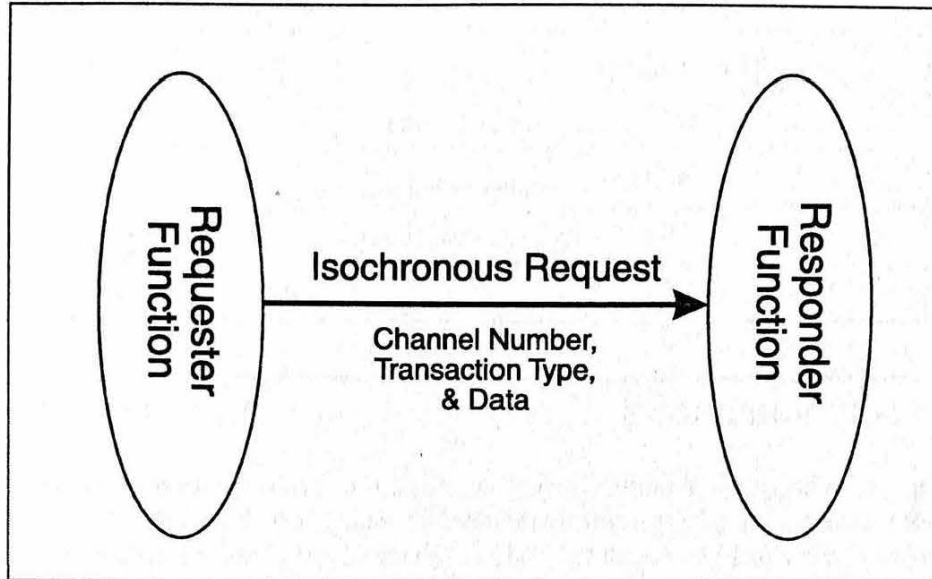


This basic communications model is defined by the CSR architecture. The serial bus provides further granularity in the transaction process, which includes verification of packet delivery, three protocol layers, and related services that perform specific functions during the process of transferring data 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)

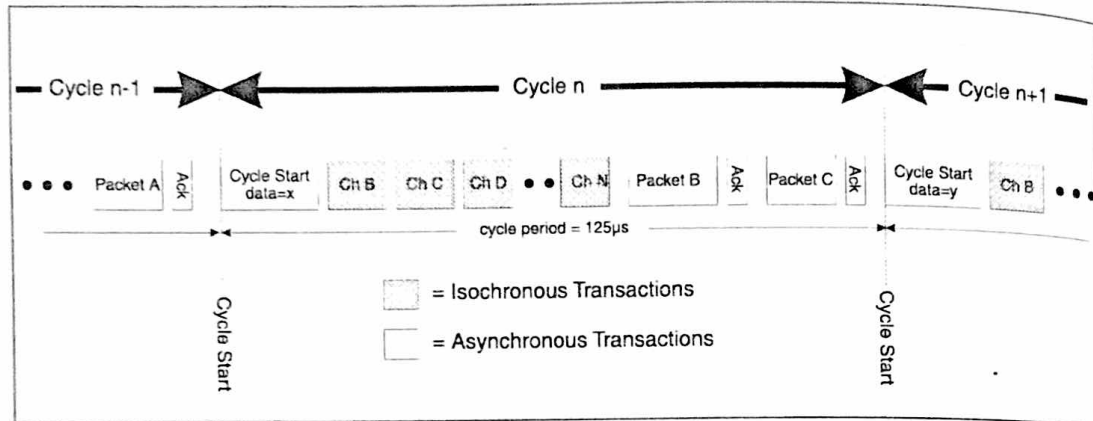
Prior to transferring data, the transmitting node must obtain ownership of the 1394 bus via an arbitration mechanism. This ensures that only one node at a time is transmitting data over the wire.

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 125 μ s intervals, called cycles. Details regarding these transaction types and their bandwidth allocation are discussed below.

FireWire System Architecture

Figure 3-3: Isochronous and Asynchronous Transactions Share Bus Bandwidth



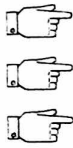
Asynchronous

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

	Cable Speed	Maximum Data Payload Size (Bytes)
	100Mb/s	512
	200Mb/s	1024
	400Mb/s	2048
	800Mb/s	4096
	1.6Gb/s	8192
	3.2Gb/s	16384

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.

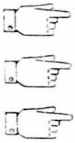
Each isochronous application must also obtain the necessary bus bandwidth that it requires for its transfer. To ensure that sufficient bus bandwidth is available, applications wishing to perform isochronous transfers must request the needed bandwidth from the isochronous resource manager node. Bus bandwidth 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 125 μ s cycle. Up to 80% (100 μ s) 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

FireWire System Architecture

amount of bandwidth available when the allocation request is made. (See "Bus Bandwidth Allocation" on page 339.)

Table 3-2: Maximum Data Block Size for Isochronous Transfers

	Cable Speed	Maximum Data Payload Size (Bytes)
	100Mb/s	1024
	200Mb/s	2048
	400Mb/s	4096
	800Mb/s	8192
	1.6Gb/s	16384
	3.2Gb/s	32768

The Protocol Layers

Four protocol layers are defined to simplify the implementation of hardware and software. Each layer has an associated set of services defined to support communications between the application and the 1394 protocol layers, and for configuration and bus management. Figure 3-4 on page 43 illustrates the relationships between these layers for a single node.

The protocol layers consist of the:

- Bus Management layer — supports bus configuration and management activities for each node.
- Transaction layer — supports the CSR architecture request-response protocol for read, write, and lock operations related to asynchronous transfers. Note that a transaction layer exists in both the requester and responder. Note also that the transaction layer does not provide any services for isochronous transfers. Instead, isochronous transfers are driven directly by the application.
- Link layer — provides the translation of a transaction layer request or response into a corresponding packet, or subaction, to be delivered over the serial bus. This layer also provides address and channel number decoding for incoming asynchronous or isochronous packets. CRC error checking is also performed here.
- Physical layer — provides the electrical and mechanical interface required

FireWire System Architecture

The common mode output voltage at TPA (receiving port) will be detected between 1.438v and 1.665v during S200 packet transmission, and between 1.030v and 1.438v during S400 packet transmission. When no speed signaling is performed (S100 packet transmission), the common mode output voltage will be greater than 1.665v. The input voltage at the opposite end of the cable (TPB) can have up to a 0.5vdc drop due to line loss in the cable. The common mode input voltages at TPB (transmitting port) will be equal to or greater than 1.165v when no speed signaling is used. When S200 speed signaling is used the input voltage ranges from 0.935 to 1.165v and with S400 speed signaling the voltage ranges from 0.523 to 0.935v. Note that during S200 and S400 speed signaling the input voltage to the port status receiver drops below 0.8v. Unless some action is taken the PHY would detect node detachment. To prevent this from occurring, the PHY must ignore disconnect status during speed signaling.

Table 6-12: Common Mode Current for Speed Signaling

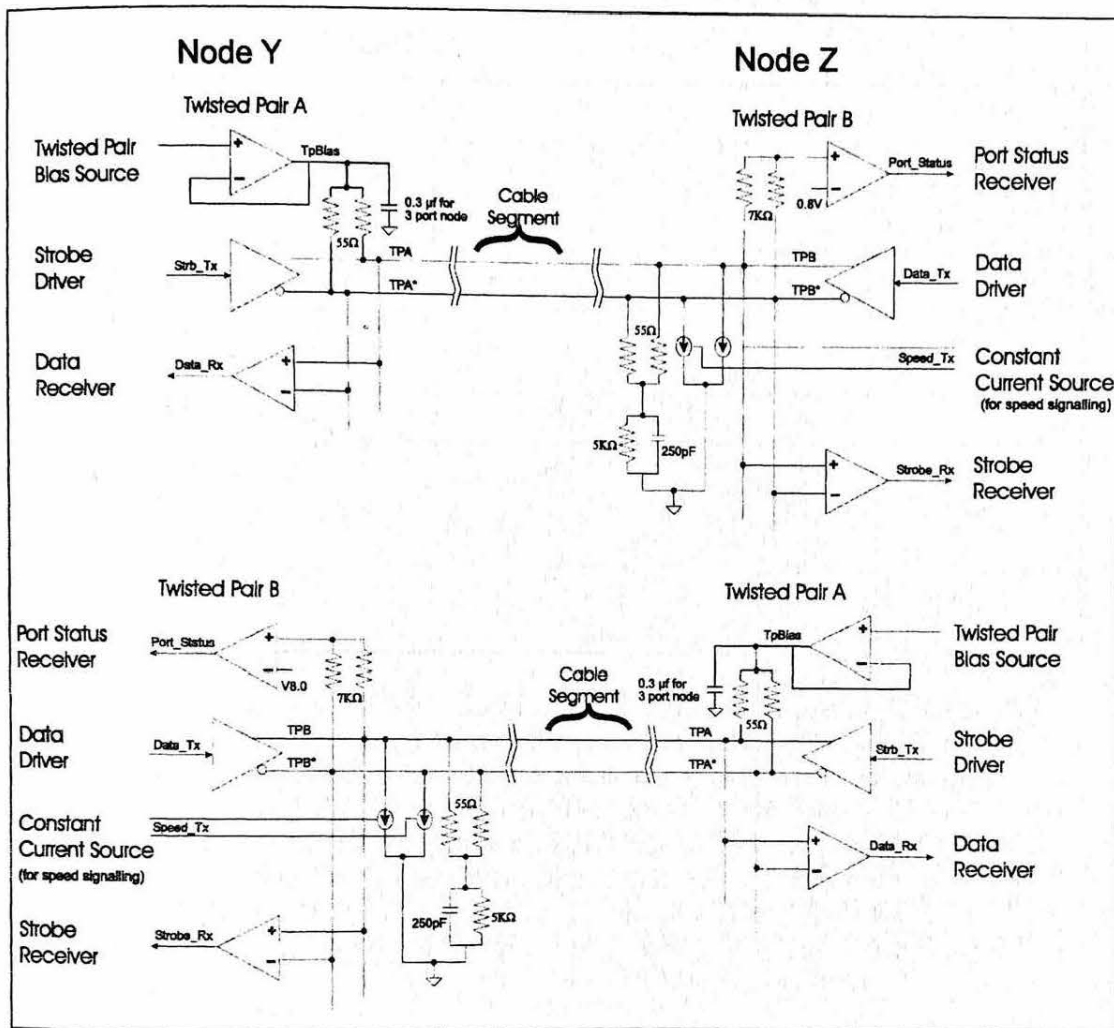
Signaling Active	Data Rate (mA)					
	Tx_Speed = 100		Tx_Speed = 200		Tx_Speed = 400	
	Max	Min	Max	Min	Max	Min
Common mode signaling off	0.44	-0.81	0.44	-0.81	0.44	-0.81
Common mode signaling on	0.44	-0.81	-2.53	-4.84	-8.10	-12.40

Data/Strobe Signaling

The actual packet is delivered via the data driver using a combination of NRZ and data-strobe encoding. Data is transferred across one signal pair while the strobe is delivered over the other pair. Single direction transfers require the use of both signal pairs, therefore bi-directional transfers are accomplished via a half-duplex implementation. During normal differential transmission of packets, the transmitter delivers the strobe via TPA and data via TPB as illustrated in Figure 6-14. Since the rate of data delivery can vary, speed information is also sent prior to packet transmission.

Chapter 6: The Electrical Interface

Figure 6-14: Data-Strobe Signaling During Packet Transmission



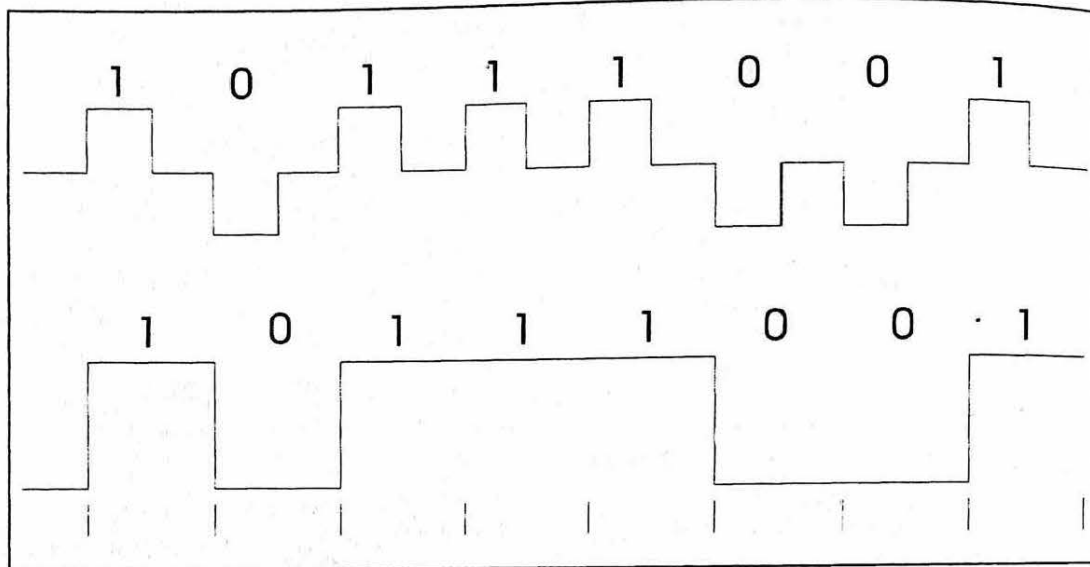
NRZ Encoding

Non-return to zero (NRZ) encoding is used to reduce the frequency component of a bipolar data stream comprised of ones and zeros. As its name implies, NRZ encoded data does not return to zero after signaling each state. Instead, it transitions only when a change from zero to one or one to zero occurs. Figure 6-15 illustrates a bipolar data stream that is not encoded followed by NRZ encoded data.

FireWire System Architecture

Data-Strobe Encoding

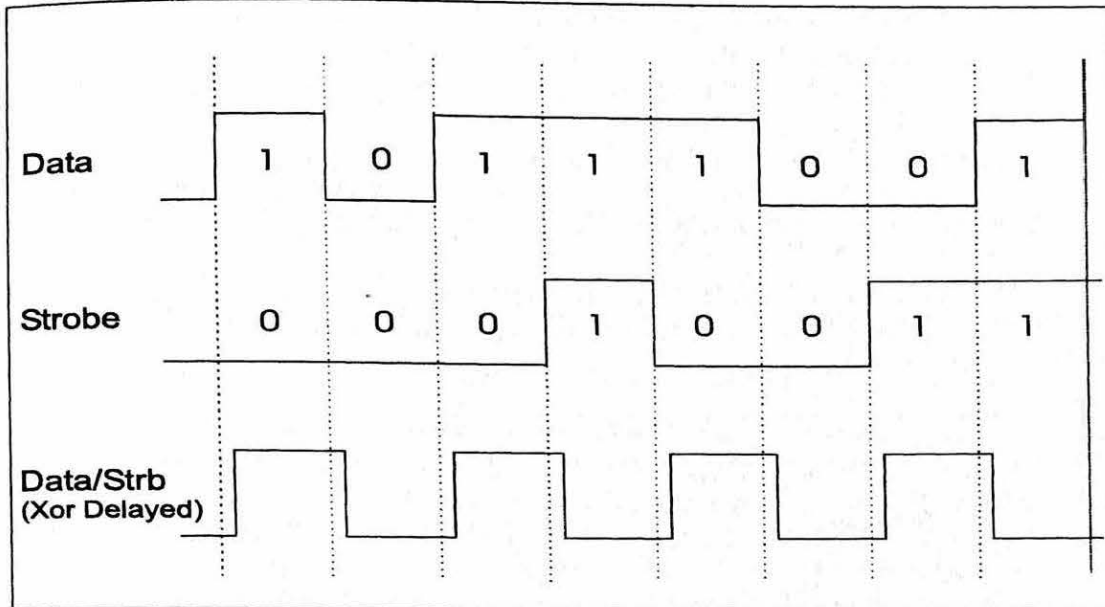
Figure 6-15: NRZ Data Stream



NRZ data is transmitted via TPB and detected by the differential data receiver on TPA. The data stream is accompanied by a strobe signal that provides a means of reconstructing a clock signal that can be used to synchronize the incoming data stream. Strobe is transmitted via TPA and received by the differential strobe receiver on TPB. The data-strobe encoding improves the transmission characteristics of data sent over the cable. Figure 6-16 illustrates an example data stream and the accompanying strobe. Note that a clock signal can be constructed by performing an exclusive OR on the data and strobe.

Chapter 6: The Electrical Interface

Figure 6-16: Data-Strobe Encoding/Decoding



Gap Timing

Between packet transfers the bus returns to the idle state for a period of time, called an interpacket gap. For example, when a read transaction begins, a request packet is sent to the target device and the target node responds by returning data to the initiating node. Sufficient turn-around time must be given for the target node to return data. A variety of idle times (called gaps) are employed by the IEEE 1394 implementation. These gaps include:

- **Acknowledge Gap**—An acknowledge gap is the bus idle time that exists between the end of an asynchronous packet (request or response) and the start of the acknowledgment packet. Note that the request or response packet and the subsequent acknowledge packet are performed as atomic operations on the bus. No other bus activity is allowed to take place during this interval. This gap must be shorter than the subaction gap which is used to notify a node that it may begin arbitration for ownership of the bus.
- **Isochronous Gap**—The period of bus idle time before the start of arbitration for an isochronous channel subaction. This gap is also shorter than the subaction gap; thereby providing a higher arbitration priority for isochronous

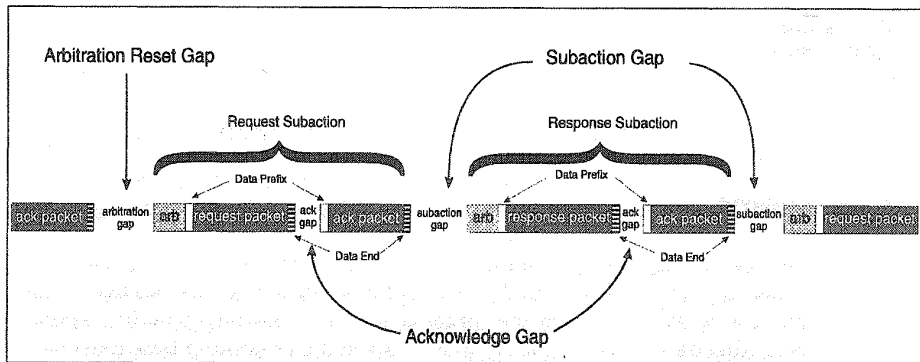
FireWire System Architecture

transactions versus asynchronous transactions which must wait for a longer period of idle time before arbitrating.

- **Subaction Gap**—The bus idle time before the start of arbitration for an asynchronous subaction. The subaction gap is not present between the request and response subactions of a concatenated transaction. The width of this gap can be tuned depending on the maximum number of cable hops in the physical topology.
- **Arbitration Reset Gap**—The period of bus idle prior to the beginning of a fairness interval. The width of this gap can be tuned depending on the maximum number of cable hops in the physical topology.

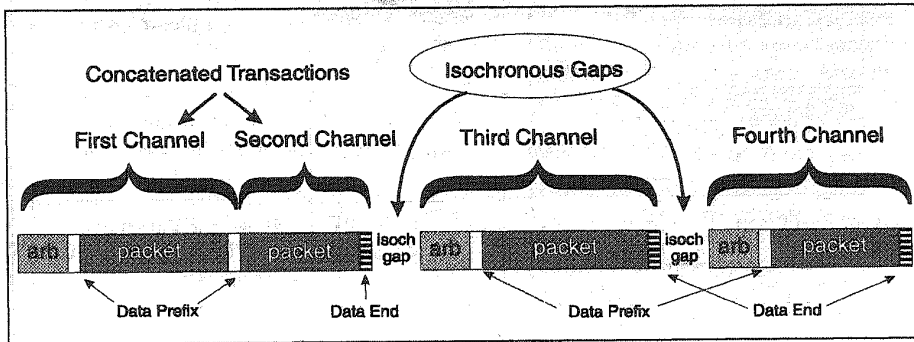
Figure 6-17 illustrates the gaps associated with asynchronous transactions. In some cases subactions can be concatenated, thereby eliminating the subaction gap.

Figure 6-17: Split Asynchronous Transaction with Gaps Defined



Isochronous gaps occur between isochronous transfers that are not concatenated as illustrated in Figure 6-18.

Figure 6-18: Isochronous Transaction with Gaps Defined



Note that the various gaps are of differing lengths as defined in Table 6-13. Gap count is related to the number of cable hops between nodes, and can be adjusted to reduce the amount of idle time on the bus. The maximum cable hop value is defined as the largest number of cables that must be traversed when any two nodes residing on the same 1394 bus communicate with one another. The default gap count value used to calculate the subaction and arbitration reset gap timing is 63 and is sufficient to accommodate 32 cable hops. This is a conservative gap count since the specification allows a maximum of 16 cable hops on a single 1394 bus.

Table 6-13: Gap Timing

Gap Type	Detection Time (measured at responding node)		Description
	Minimum	Maximum	
Acknowledge gap	0.04μs	0.05μs	~4/base rate—time between end of packet and return of acknowledgment (at port transmitting the Ack. packet)
Isochronous gap	0.04μs	0.05μs	~4/base rate—bus idle time between nonconcatenated isochronous transactions

8

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 125 μ s 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.

FireWire System Architecture

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

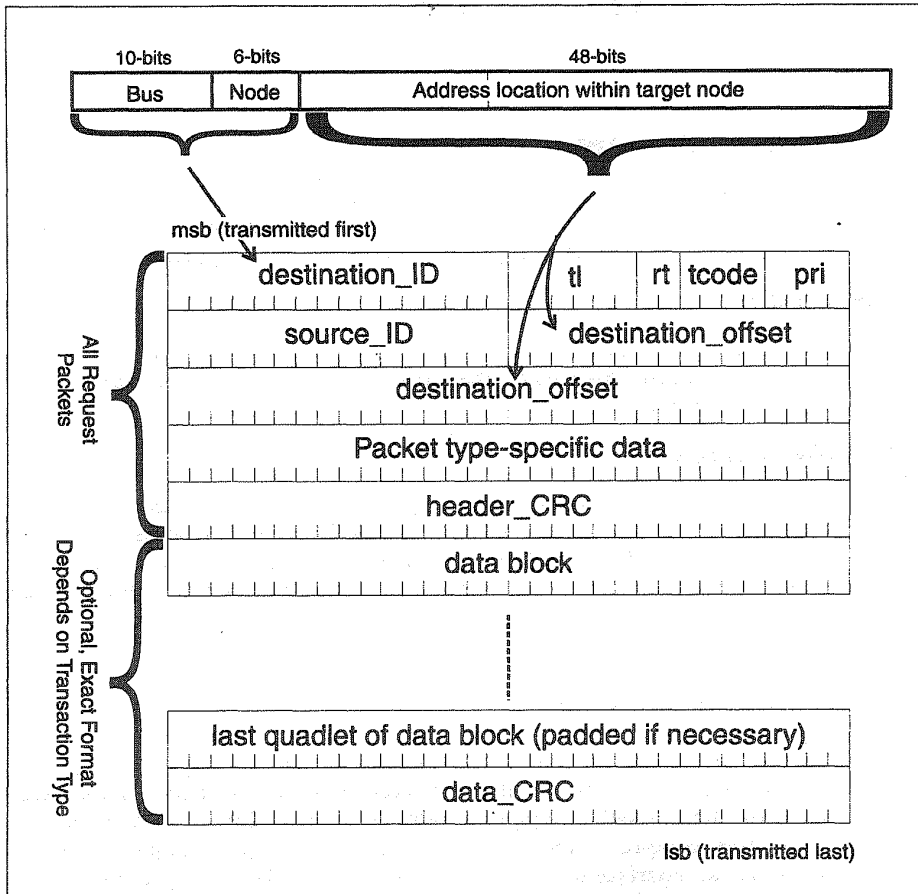


Table 8-1: Asynchronous Transaction Codes




Transaction Name	Code (Hex)
Write Request for data quadlet	0
Write Request for data block	1
Write Response	2
Reserved	3
Read request for data quadlet	4
Read request for data block	5
Read response for data quadlet	6
Read response for data block	7
Cycle start	8
Lock request	9
Asynchronous Streaming Packet	A
Lock response	B
Reserved	C
Reserved	D
Used internally by some link designs. Will not be standardized	E
Reserved	F

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 125 μ s 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.

FireWire System Architecture

Table 8-2: Maximum Data Payload for Asynchronous Packets

	Cable Speed	Maximum Data Payload Size (Bytes)
	100Mb/s	512
	200Mb/s	1024
	400Mb/s	2048
	800Mb/s	4096
	1.6Gb/s	8192
	3.2Gb/s	16384

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).

Chapter 8: Asynchronous

Figure 8-2: Write Request — Quadlet Format

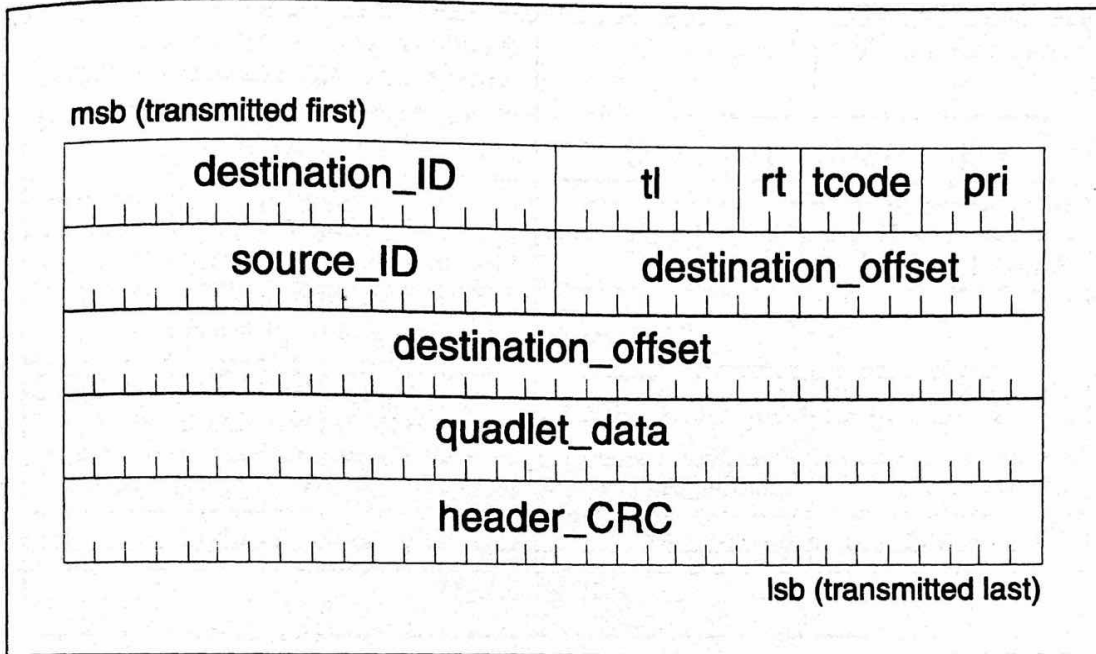


Table 8-3: Write Request — Quadlet Packet Components

Name	Abbrev.	Description
Destination Identifier	destination_ID	Combination of the bus address and physical ID of the node. Contains the address of the requesting node.
Transaction label	tl	A label specified by the requester that identifies this transaction. This value, if used, is returned in the response packet.
Retry code	rt	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

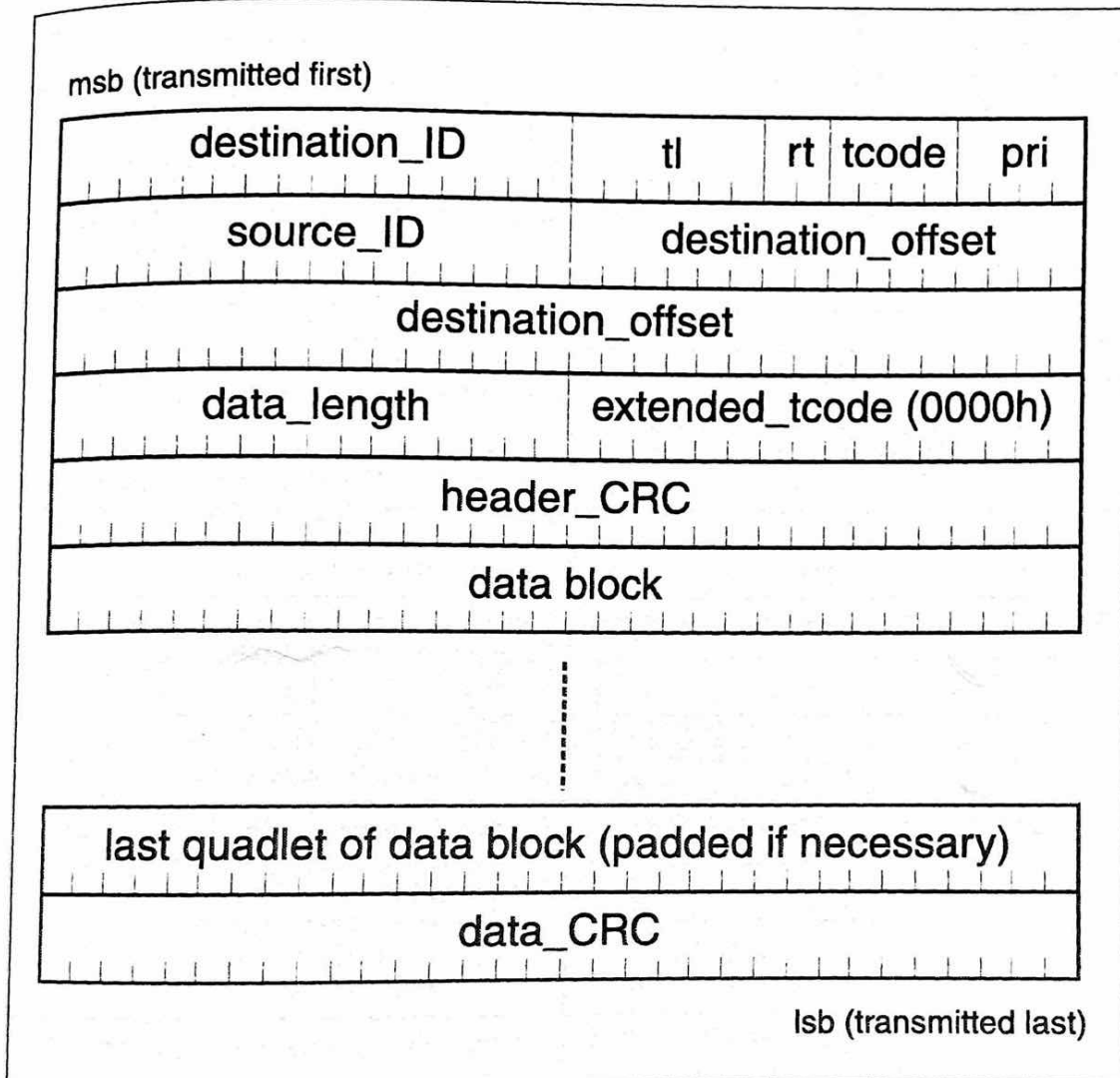
FireWire System Architecture

Table 8-3: Write Request — Quadlet Packet Components (Continued)

Name	Abbrev.	Description
Transaction Code	tcode	A code value of zero defines write request for quadlet data, thereby defining the packet format illustrated in Figure 8-2 on page 169.
Priority	pri	Not used in cable environment.
Source Identifier	source_ID	Identifies the node that is sending this packet by specifying the bus that it resides on and its physical ID.
Destination Offset	destination_offset	Specifies the address location within the target node that is being accessed.
Quadlet Data	quadlet_data	Data being delivered to the target node.
Header CRC	header_CRC	CRC value for the header.

Chapter 8: Asynchronous

Figure 8-3: Write Request — Block Format



FireWire System Architecture

Table 8-4: Write Request — Block Packet Components

Name	Abbrev.	Description
Destination Identifier	destination_ID	Combination of the node address and physical ID of the node. Uniquely identifies a node. Contains the address of the requesting node.
Transaction label	tl	A label specified by the requester that identifies this transaction. This value, if used, is returned in the response packet.
Retry code	rt	This code specifies whether this packet is an attempted retry and defines the retry protocol to be followed by the target node.
Transaction Code	tcode	A code value of one defines write request for block of data, thereby defining the packet format illustrated in Figure 8-3 on page 171.
Priority	pri	Not used in cable environment.
Source Identifier	source_ID	Identifies the node that is sending this packet by specifying the bus that it resides on and its physical ID.
Destination Offset	destination_offset	Specifies the address location within the target node that is being accessed.
Data Length	destination_length	Specifies the amount of the data being sent in the data field of this packet. Maximum size in bytes is as follows: <ul style="list-style-type: none"> • 512 @ 100Mb/s • 1024 @ 200Mb/s • 2048 @ 400Mb/s
Extended Transaction code	extended_tcode	Reserved during write request packets.
Header CRC	header_CRC	CRC value for the header.
Data Field	data field	Contains the data being transferred to the target device. Data must be padded with zeros, if not an even 4 bytes. Amount of data is specified in the data_length field.
Data CRC	data_CRC	CRC value for the data field.

Chapter 8: Asynchronous

Figure 8-4: Write Response Packet Format

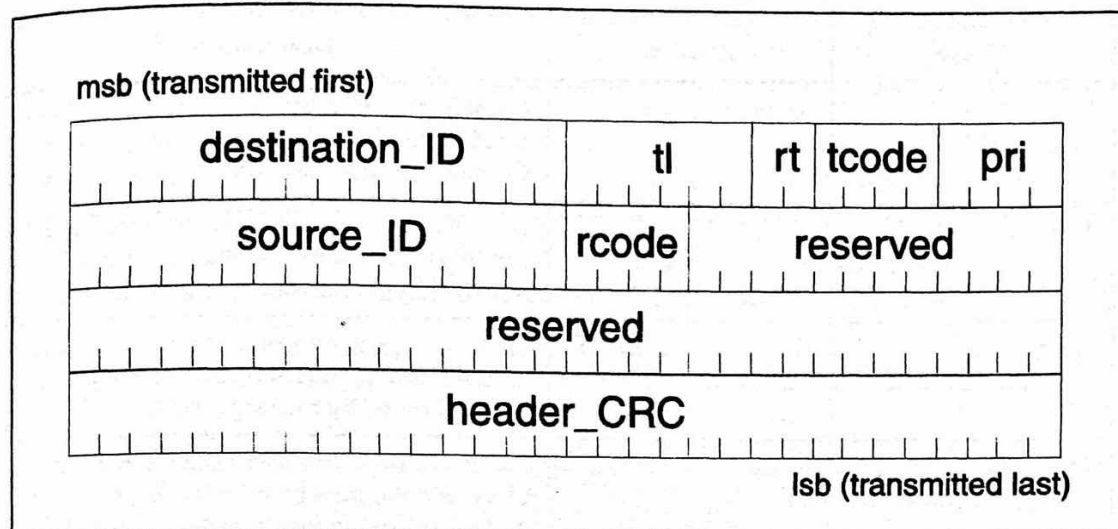


Table 8-5: Write Response Packet Components

Name	Abbrev.	Description
Destination Identifier	destination_ID	Combination of the bus address and physical ID of the node. Uniquely identifies a node. Contains the address of the node that is to receive the response packet.
Transaction label	tl	Contains the value sent by the requester for this transaction.
Retry code	rt	This code specifies whether this packet is an attempted retry and defines the retry protocol to be followed by the target node.
Transaction Code	tcode	A code value of two defines write response for either type of write request, and defines the packet format illustrated in Figure 8-4.

FireWire System Architecture

Table 8-5: Write Response Packet Components (Continued)

Name	Abbrev.	Description
Priority	pri	Not used in cable environment.
Source Identifier	source_ID	Identifies the node that is sending this response packet by specifying the bus that it resides on and its physical ID.
Response Code	rcode	Specifies the result of this transaction. Refer to Table 8-14 on page 191 for code definitions.
Header CRC	header_CRC	CRC value for the header.

Asynchronous Stream Packet

The 1394a supplement defines an asynchronous stream packet (tcode = A) that has the same format and characteristics as the isochronous packet, except that it occurs during a fairness interval and not during the isochronous bus interval. Format of this packet is illustrated in Figure 8-5 on page 175. The asynchronous stream packet is also referred to as a loose isochronous packets. This terminology derives from PHY implementations based on the 1995 specification. Some PHYs do not allow this packet to be transmitted during the asynchronous bus time (strict isochronous packets), while other implementations do allow the packet to be sent during the asynchronous time (loose isochronous packets).

The asynchronous stream transaction allows applications to perform data streaming when guaranteed latency is of little concern. This eliminates the requirement to allocate isochronous bus bandwidth (a limited resource) in order to perform their transaction. However, a channel number must be obtained in order to perform an asynchronous stream transaction. Maximum packet size is constrained to the values specified for all asynchronous packets as listed in Table 8-2 on page 168. This type of transaction could be used by devices that support broadcast and/or multicast operations, but do not require the transmission of real-time data.

Note that the link layer of a node wishing to transmit an asynchronous streaming packet uses a fair or priority arbitration service. When the PHY has obtained bus ownership, the link transmits the asynchronous streaming packet. Note that there is no acknowledge packet or response returned by the targeted node.

9

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 125 μ s 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 (known 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

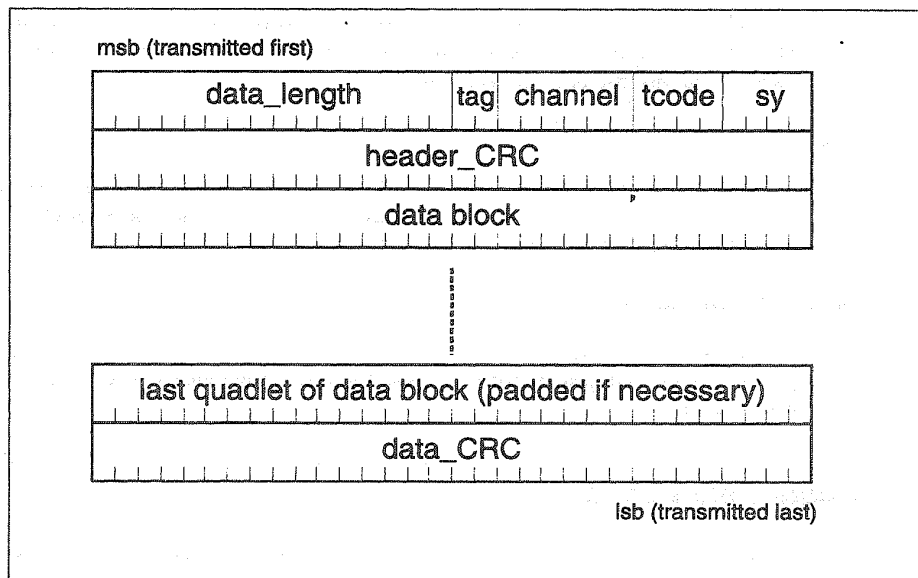


Table 9-1: Isochronous Stream Packet Components

Component Name	Abbreviation	Description
Data Length	data_length	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.
Isoch Data Format Tag	tag	The value of 00b indicates that the isochronous data is unformatted. All other values are reserved.
Isoch Channel Number	Channel	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.
Transaction code	tcode	The transaction code for an isochronous data block transaction is Ah.
Synchronization Code	sy	Application specific.
Header CRC	header_CRC	CRC value for header.
Data Block Payload	data_field	Data to be transferred by the talker. Last quadlet of data field must be padded with zeros if necessary.
Data Block CRC	data_CRC	CRC value for the data field.

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 100 μ s. 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.

FireWire System Architecture

Table 9-2: Maximum Data Payload for Isochronous Packets

	Cable Speed	Maximum Data Payload Size (Bytes)
	100Mb/s	1024
	200Mb/s	2048
	400Mb/s	4096
☞	800Mb/s	8192
☞	1.6Gb/s	16384
☞	3.2Gb/s	32768

Isochronous Transaction Summary

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

Figure 9-2: Non-Concatenated Isochronous Transactions

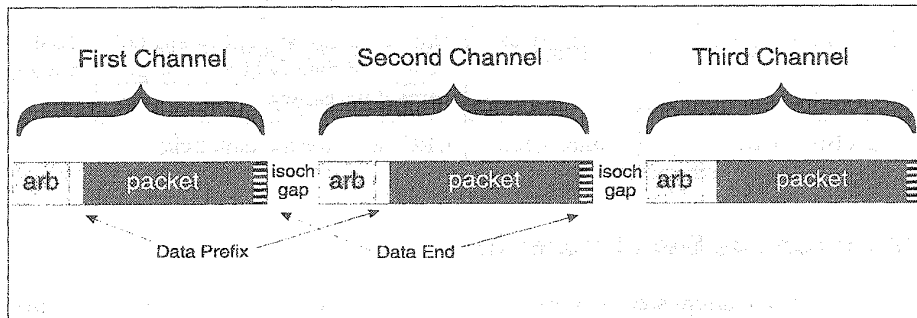
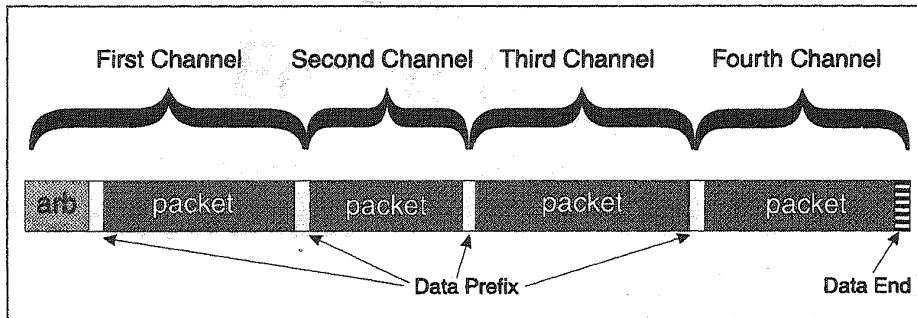


Figure 9-3: Concatenated Isochronous Transactions - If Sent by Same Node



FireWire System Architecture

FireWire System Architecture

Powering the Link

During bus configuration the link may be powered off. However, nodes that are capable of becoming bus managers or isochronous resource managers must have their link layer powered. The bus manager or isochronous resource manager (when the bus manager node is not present) must apply power to other nodes that have their link powered off during bus configuration.

Each node reports whether its link is powered during configuration when it sends its self-ID packet. The bus manager capable and resource manager capable nodes must save the self-ID packets so they can perform bus management functions in the event that they are chosen. Following bus configuration, the node performing power management must ensure that each node's link is powered by sending it a link-on packet.

Packet Transmission

When a functional device within a node wishes to perform a transaction, the associated application issues a transaction request to the link layer chip. The link layer then uses the PHY/link interface to forward the request to the PHY. The request is issued to the PHY chip via the LReq signal line. Once the request has been issued, the PHY must arbitrate for control of the IEEE 1394 bus and grant use of the PHY/Link interface so that the necessary packets can be sent to the PHY for transmission over the 1394 bus.

Generally, the request may be issued at any time to the PHY. However, to ensure proper operation of the link/PHY interface, certain rules of conduct are defined that limit when a link request can be issued. These rules depend on whether the link or PHY currently owns the interface and how the interface is currently being used.

Link Issues Request

The link requests access to the serial bus via the LReq line by sending a serial stream of bits to the PHY. These bits define the type of request being made and the speed at which the packet must be delivered. Note that the link always owns the unidirectional LReq line and can issue a request asynchronously to the PHY. However, certain requests can only be issued at particular times (discussed later in the section). When a transaction request is issued to the PHY it must arbitrate for control of the serial bus. Once ownership of the 1394 bus has

FireWire System Architecture

Table 11-8: Link Requests and Corresponding PHY Behavior and Link Actions (Continued)

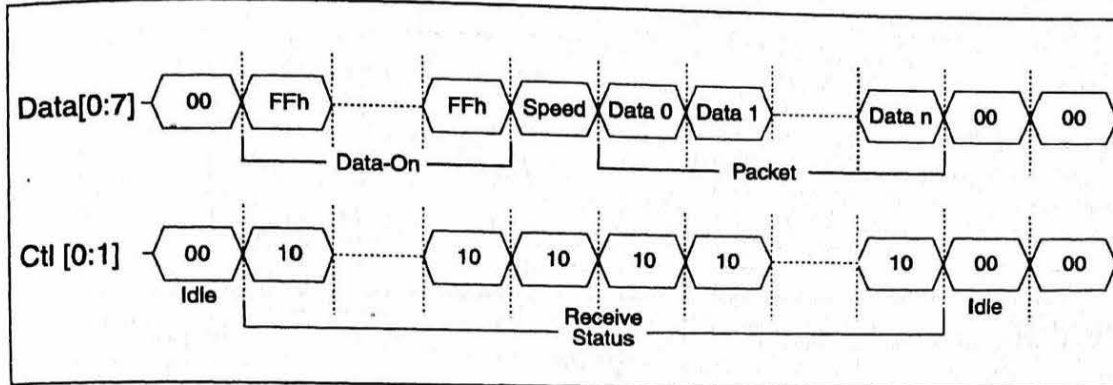
Request Type	C _B States During or After LReq transmission	PHY Behavior	Link Action
Isochronous	Idle		Monitor Ctl for the next change of state.
	Status	Request is serviced, unless status indicates a subaction gap event. This indicates that the isochronous bus time has expired, reflecting an error condition.	Link continues to monitor Ctl line, unless request is discarded.
	Receive	PHY retains the request.	Link monitors Ctl lines.
	Grant	The PHY has won arbitration and is ready to receive a packet.	Link transmits the packet.
	Transmit, Idle (interface driven by Link)	Request is retained by PHY.	Monitor Ctl lines after releasing control of interface to the PHY.

Receiving Packets

When a packet is being transmitted by another node the PHY detects the "Data-On" state. It then signals a receive condition on Ctl[0:1] and delivers all ones on the data lines. The PHY signals the beginning of the packet by sending the speed code over the data lines, which is immediately followed by packet contents. Note that the CRC check does not include the speed code information. Receive signaling is illustrated in Figure 11-4 on page 237.

Chapter 11: Link to PHY Interface

Figure 11-4: Sequence and Format of Data Lines During Packet Receive State



Speed of PHY to Link Data Transmission

When a packet is received by the PHY it must transfer the packet to the Link. The PHY transmits the packet to the link at the received speed. For example, if the packet is received at 100Mb/s only two of the data lines are used, whereas, if the packet is received at 400Mb/s all eight data lines are used. The PHY indicates the start of a receive packet by sending a speed code over the data lines. This serves to notify the link that a packet is being transmitted and the speed at which it will be sent (i.e. the number of data lines that will be used). Table 11-9 shows the speed code signaling used by the PHY when starting packet transfer over the PHY/Link data lines. Note that if packet speed is 100Mb/s only two bits are needed to define the speed (00b), while 200Mb/s transfers use four bits and 400Mb/s and greater use all eight lines.

A PHY may support a higher transfer rate than the link. In this case the link will recognize that it does not support the incoming packet when it detects the speed code and consequently will ignore the packet transmission. The link ignores all interface traffic until it detects the next idle state.

Table 11-9: Speed Signaling PHY to Link

D[0:7]	Data Rate (Mb/s)
00xxxxxb	100
0100xxxxb	200
01010000b	400

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

	SELF_ID_GRANT	DATA_PREFIX	IDENTIFICATION_DONE
TPA (Strobe)	Z	0	1
TPB (Data)	0	1	Z

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.

17

Cycle Master

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 125 μ s intervals. The root node is responsible for specifying the 125 μ s interval and marks the beginning of the next series of isochronous transactions by broadcasting a cycle start packet as illustrated in Figure 17-1.

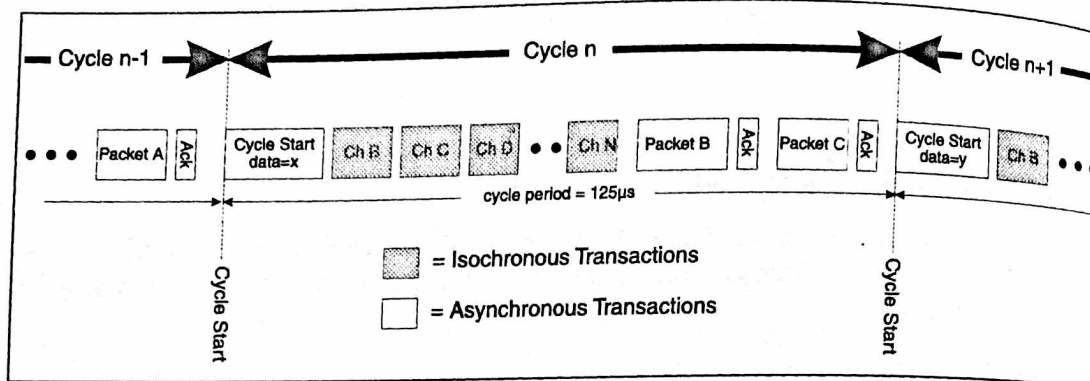
Determining and Enabling the Cycle Master

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

FireWire System Architecture

Figure 17-1: Isochronous Transactions Performed Every 125 μ s



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 125 μ s). 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 125 μ s 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

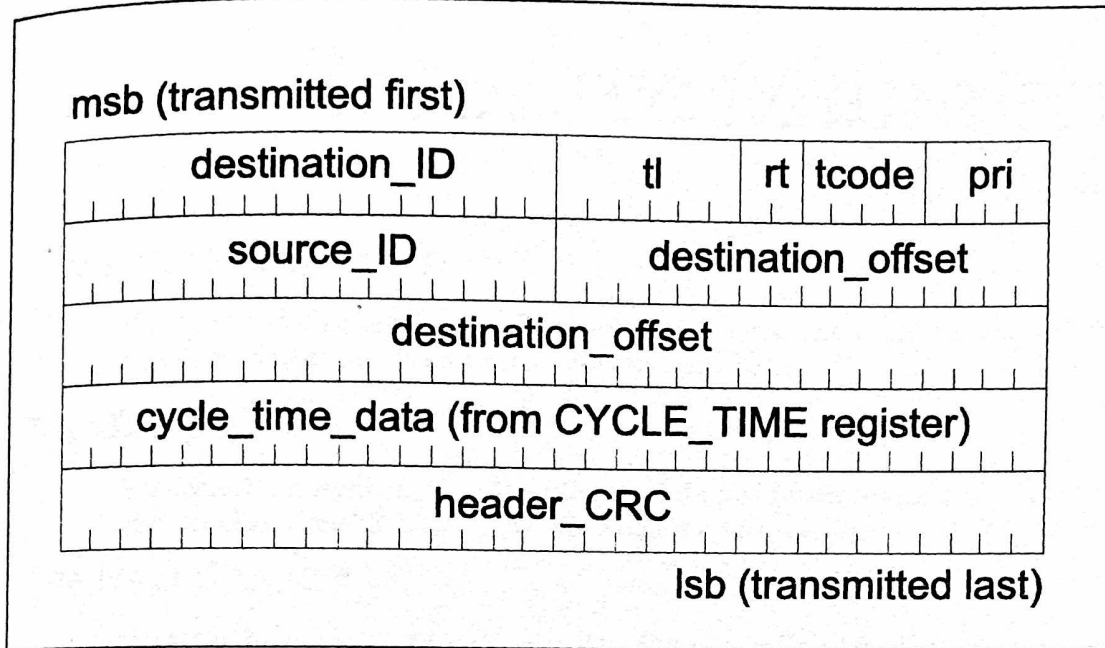
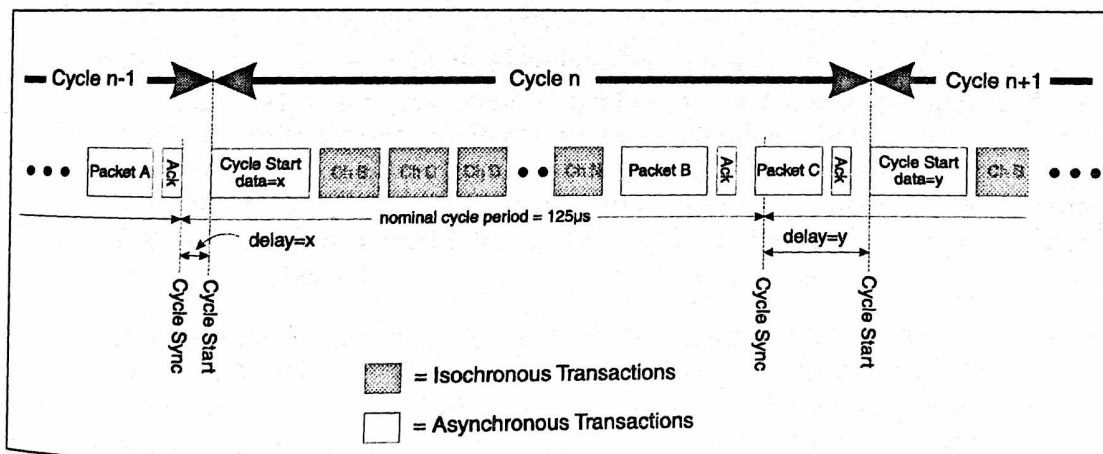


Figure 17-3: Cycle Time Variation Included in Cycle Start Packet



Appendix: Example 1394 Chip Solutions

Overview

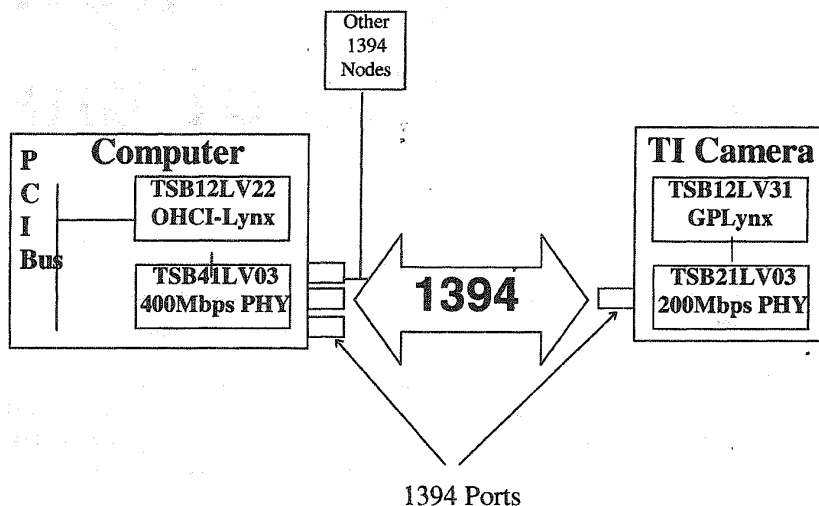
Below you will see two of the most popular 1394 applications today; a 1394 enabled PC and a 1394 Digital camera. Texas Instrument's extensive 1394 portfolio includes all the silicon necessary to implement both applications. For more detailed information regarding TI's 1394 solutions, please visit our website at <http://www.ti.com/sc/1394>.

In this Appendix we will explain 1394 in the PC environment and in a TI 1394 camera by showing you a detailed block diagram and providing you an overview of the 1394 silicon used in each.

1394 in the PC

TI has many Link and Physical Layer Controllers necessary to implement 1394 in a PC environment; however, in this Appendix we will focus on TI's newest Link Layer TSB12LV22 and Physical Layer TSB41LV03.

FireWire System Architecture



TSB12LV22 / OHCI-Lynx

The industry's first Open Host Controller Interface (OHCI) Link Layer controller will revolutionize the way we use our computers. The OHCI specification provides the host PC a way to communicate with high-speed peripherals via a single OHCI driver. With an OHCI link layer device, the system software will not require a separate driver. This will enable add-in cards in the host OS as well as boot support in the PC BIOS with one Driver.

FEATURES

- Designed to IEEE1394 Open Host Controller Interface Specification
- IEEE1394-1995 Compliant, and Compatible with 1394A
- Compliant To Latest PCI Specification, Revision 2.1 and is PCI 2.2 ready
- PCI Power Management Compliant
- 3.3 Volt Core Logic with Universal PCI Interface Compatible with 3.3 Volt and 5 Volt PCI Signaling Environments

- Supports Serial Bus Data Rates of 100, 200, and 400Mbit/s
- Provides Bus Hold Buffers on Physical I/F for Low Cost Single Capacitor Isolation
- Supports Physical Write Posting of up to Three Outstanding Transactions
- Serial ROM Interface Supports 2-Wire Devices
- Supports External Cycle Timer Control for Customized Synchronization
- Implements PCI Burst Transfers and Deep FIFOs To Tolerate Large Host Latency
- Provides Four General Purpose I/Os
- Fabricated in Advanced Low-Power CMOS Process
- Packaged in 100 TQFP

OVERVIEW

The Texas Instruments OHCI-Lynx is a PCI-to-1394 host controller compatible with the latest PCI, IEEE1394, and 1394 OHCI Specifications. The chip provides the IEEE1394 link function, and is compatible with serial bus data rates of 100, 200, and 400 Mbits per second.

As required by the 1394 Open Host Controller Interface Specification, internal control registers are memory mapped and non-prefetchable. The PCI configuration header is accessed through configuration cycles specified by PCI, and provides Plug and Play compatibility. Furthermore, the OHCI-Lynx is compliant with the PCI Power Management Specification, per Microsoft's PC '98 requirements.

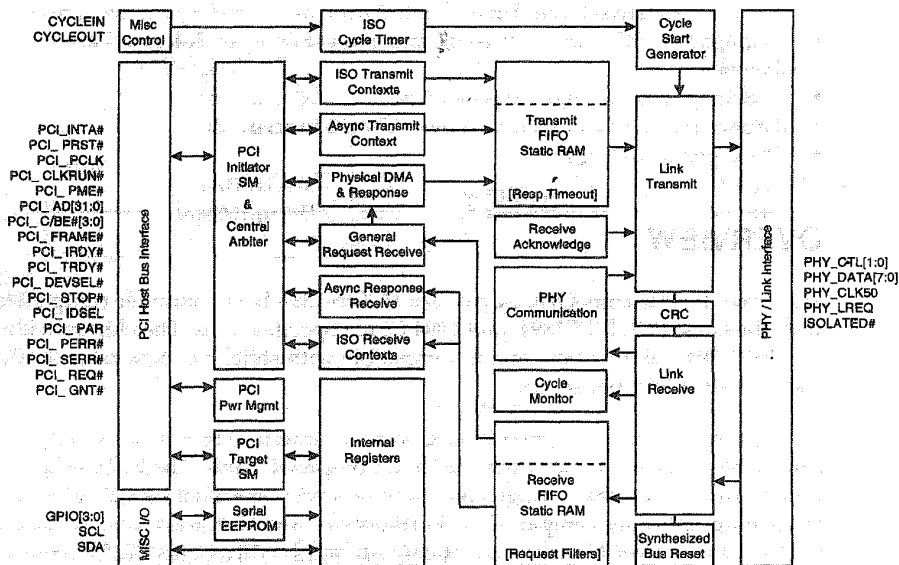
The OHCI-Lynx design provides PCI bus master bursting, and is capable of transferring a cacheline of data at 132Mbytes/sec after connection to the memory controller. Since PCI latency can be rather large even on a PCI Revision 2.1 system, deep FIFOs are provided to buffer 1394 data.

Physical write posting buffers are provided to enhance serial bus performance, and multiple isochronous channels are provided for simultaneous operation of real-time applications. The OHCI-Lynx also provides bus holding buffers on the phy/link interface for simple and cost effective single capacitor isolation.

FireWire System Architecture

BLOCK DIAGRAM

A simplified block diagram of the OHCI-Lynx is provided below.



TSB41LV03

TI's newest Physical Layer is the first of many 400Mbps Physical Layers from Texas Instruments. The TSB41LV03 provides the bandwidth necessary to power the most demanding applications such as hard disk drives, hosts and multimedia peripherals. The TSB41LV03 has three 1394 ports, with the complete family supporting 2, 3, 4, and 6 port versions.

FEATURES

- Supports IEEE 1394-1995 Standard for High Performance Serial Bus and 1394.A
- Provides Three Fully Compliant Cable Ports at 100/200/400 Megabits per Second (Mbit/s)
- Cable Ports Monitor Line Conditions for Active Connection to Remote Nodes
- Device Power-down Feature to Conserve Energy in Battery Powered Applications
- Inactive Ports Disabled to Save Power
- Logic Performs System Initialization and Arbitration Functions
- Encode and Decode Functions Included for Data-Strobe Bit Level Encoding
- Incoming Data Resynchronized to Local Clock
- Single 3.3 Volt Supply Operation
- Interface to Link Layer Controller Supports Optional Annex J Electrical Isolation and TTMBus-Holder Isolation
- Data Interface to Link-Layer Controller Provided Through 2/4/8 Parallel Lines at 50 MHz
- 25-MHz Crystal Oscillator and PLL Provide Transmit, Receive Data at 100/200/400 Megabits per Second, and Link-Layer Controller Clock at 50 MHz
- Interoperable with 1394 Link-Layer Controllers Using 5-V Supplies
- Interoperable Across 1394 Cable with Physical Layers (PHY) Using 5-V Supplies
- Mode Power Class Information Signaling for System Power Management
- Cable Power Presence Monitoring
- Separate Cable Bias and Driver Termination Voltage Supply for Each Port
- Multiple Separate Package Terminals Provided for Analog and Digital Supplies and Grounds
- High Performance 80 Pin TQFP (PFP) Thermally Enhanced Package
- Register bits give software control of contender bits, power class bits, and 1394.A features

TSB41LV03 OVERVIEW

The TSB41LV03 provides the analog transceiver functions needed to implement a three port node in a cable-based IEEE 1394-1995 or IEEE-1394.A network. Each cable port incorporates two differential line transceivers. The transceivers include circuitry to monitor the line conditions as needed for determining connection status, for initialization and arbitration, and for packet reception and transmission. The TSB41LV03 is designed to interface with a Link Layer Controller (LLC), such as the TSB12LV22, TSB12LV21A, TSB12LV31, or TSB12C01A.

FireWire System Architecture

The TSB41LV03 requires an external 24.576 MHz crystal or crystal oscillator. The crystal oscillator drives an internal phase-locked loop (PLL), which generates the required 393.216 MHz reference signal. This reference signal is internally divided to provide the clock signals used to control transmission of the outbound encoded Strobe and Data information. A 49.152 MHz clock signal is supplied to the associated LLC for synchronization of the two chips and is used for resynchronization of the received data. The power-down (PD) function, when enabled by asserting the PD terminal high, stops operation of the PLL.

The TSB41LV03 supports an optional isolation barrier between itself and its LLC. When the /ISO input terminal is tied high, the LLC interface outputs behave normally. When the /ISO terminal is tied low, internal differentiating logic is enabled, and the outputs are driven such that they can be coupled through a capacitive or transformer galvanic isolation barrier as described in IEEE 1394-1995 Annex J. To operate with TI Bus Holder isolation the /ISO terminal must be tied high.

Data bits to be transmitted through the cable ports are received from the LLC on two, four or eight parallel paths (depending on the requested transmission speed) and are latched internally in the TSB41LV03 in synchronization with the 49.152 MHz system clock. These bits are combined serially, encoded, and transmitted at 98.304/196.608/392.216 Mbit/s as the outbound data-strobe information stream. During transmission, the encoded data information is transmitted differentially on the TPB cable pair(s), and the encoded strobe information is transmitted differentially on the TPA cable pair(s).

During packet reception the TPA and TPB transmitters of the receiving cable port are disabled, and the receivers for that port are enabled. The encoded data information is received on the TPA cable pair, and the encoded strobe information is received on the TPB cable pair. The received data-strobe information is decoded to recover the receive clock signal and the serial data bits. The serial data bits are split into two, four or eight bit parallel streams (depending upon the indicated receive speed), resynchronized to the local 49.152 MHz system clock and sent to the associated LLC. The received data is also transmitted (repeated) on the other active (connected) cable ports.

Both the TPA and TPB cable interfaces incorporate differential comparators to monitor the line states during initialization and arbitration. The outputs of these comparators are used by the internal logic to determine the arbitration status. The TPA channel monitors the incoming cable common-mode voltage. The value of this common-mode voltage is used during arbitration to set the speed of the next packet transmission. In addition, the TPB channel monitors the incoming cable common-mode voltage on the TPB pair for the presence of the

remotely supplied twisted-pair bias voltage. The presence or absence of this common-mode voltage is used as an indication of cable connection status. The cable connection status signal is internally debounced in the TSB41LV03 and is used to initiate a bus-reset.

The TSB41LV03 provides a 1.86 Volt nominal bias voltage at the TPBIAS terminal for driver load termination. This bias voltage, when seen through a cable by a remote receiver, indicates the presence of an active connection. The value of this bias voltage has been chosen to allow interoperability between transceiver chips operating from either 5 volt nominal supplies or 3 volt nominal supplies. This bias voltage source must be stabilized by an external filter capacitor of approximately 1.0 mF.

The line drivers in the TSB41LV03 operate in a high-impedance current mode, and are designed to work with external 110W line-termination resistor networks. One network is provided at each end of a twisted-pair cable. Each network is composed of a pair of series-connected 55W resistors. The midpoint of the pair of resistors that is directly connected to the twisted pair A terminals is connected to the TPBIAS voltage terminal. The midpoint of the pair of resistors that is directly connected to the twisted-pair B terminals is coupled to ground through a parallel R-C network with recommended values of 5 KW and 250 pf. The values of the external line termination resistors are designed to meet the standard specifications when connected in parallel with the internal receiver circuits.

The driver output current, along with other internal operating currents, is set by an external resistor connected between the R0 and R1 terminals. This current setting resistor has a value of 6.3 KW, 0.5%.

The port transmitter and receiver circuitry is disabled during power-down (when the PD input terminal is asserted high), during reset (when the /RESET input terminal is asserted low), when no active cable is connected to the port, or as controlled by the internal arbitration logic. The port twisted pair bias voltage circuitry is disabled during power-down, during reset, or when the port is disabled as commanded by the LLC.

If the power supply of the TSB41LV03 is removed while the twisted pair cables are connected, then the TSB41LV03 transmitter and receiver circuitry will present a high impedance to the cable and will not load the TPBIAS voltage at the other end of the cable.

If the TSB41LV03 is being used with one or more of the ports not being brought out to a connector, the twisted pair terminals must be terminated for reliable

FireWire System Architecture

operation. For each unused port, the TPB+ and TPB- terminals must be pulled to ground through the 5 K Ω resistor of the normal termination network, thus disabling the port. The TPA+, TPA-, and TPBIAS terminals of an unused port may be left unconnected.

The TESTM, SE, and SM terminals are used to set up various manufacturing test conditions. For normal operation, the TESTM terminal should be connected to Vcc, and the SE and SM terminals should be connected to ground.

Four package terminals are used as inputs to set the default value for four configuration status bits in the self-identification packet. These terminals are hard-wired high or low as a function of the equipment design. The PC[0:2] terminals are used to indicate the power-class status for the node (the need for power from the cable or the ability to supply power to the cable). The C/LKON terminal is used as an input to indicate whether the node is a contender for bus manager. The C input corresponds to the "c" field (bit 20) in the self-id packet, and PC[0:2] inputs correspond to the "pwr" field (bits 21, 22, and 23, respectively).

The PD (power-down) terminal is provided to allow a low power mode where most of the TSB41LV03 circuits are disabled to conserve energy in battery-powered applications. When the device is powered-down it does not act as a repeater.

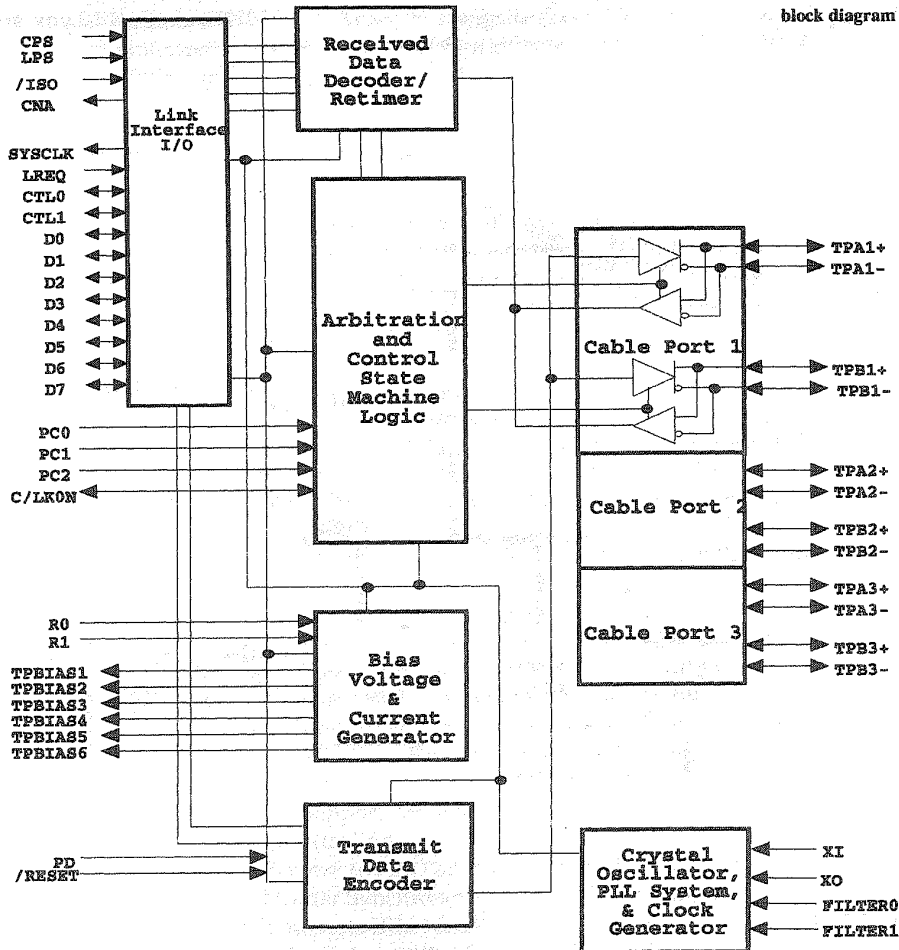
The CNA (cable-not-active) terminal provides a high output when all twisted pair cable ports are disconnected, and can be used to determine when to power-down the TSB41LV03. The CNA output is not debounced. In power-down mode, the CNA detection circuitry remains enabled.

The LPS (link power status) terminal works with the C/LKON terminal to manage the power usage in the node. The LPS signal from the LLC indicates to the PHY that the LLC is powered up and active. During LLC power-down mode, as indicated by the LPS input being low for more than 2.56 ms, the TSB41LV03 deactivates the PHY-LLC interface to save power. The TSB41LV03 will continue the necessary repeater function required for network operation during this powered-down state.

If the PHY receives a link-on packet from another node, the C/LKON terminal is activated to output a 6.15MHz signal. The LLC recognizes this signal and reactivates the powered-down portions of the LLC. After power-up of the LLC, the LLC notifies the PHY of the power-on status via the LPS terminal. The PHY confirms notification by deactivating the 6.14 MHz signal on the C/LKON terminal, then enables the PHY-link interface.

Block Diagram

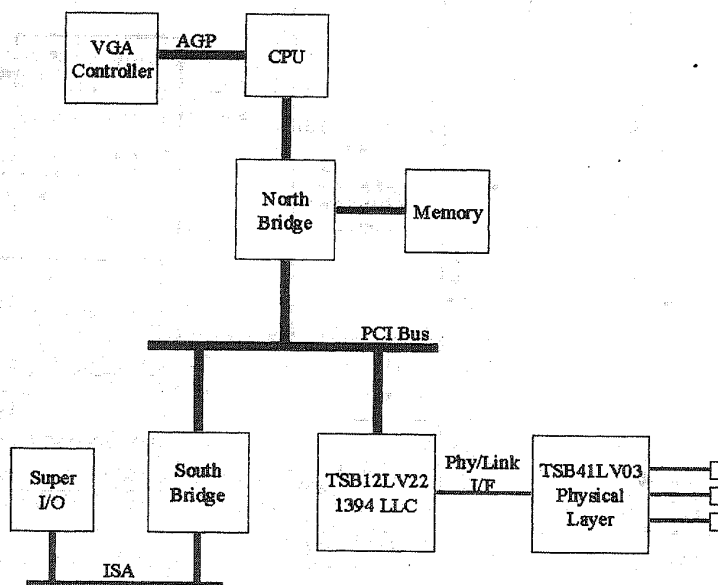
A simplified block diagram of the TSB41LV03 is shown below.



FireWire System Architecture

Putting it all Together

Below is a detailed block diagram of using the TSB12LV22/OHCI-Lynx and TSB41LV03 together to provide a 1394 interface for a PC.



Praise for the first edition:

"Finally, a book on IEEE 1394 that's easy to understand!!! MindShare's *FireWire® System Architecture: IEEE 1394* provides a consolidated learning tool for 1394 hardware and software, and confirms the statement, '1394's for the fun stuff...'"

—Phil Roden, 1394 Digital Design Manager, Texas Instruments

The FireWire® (IEEE 1394a) standard for high-speed serial bus communications has come to the fore as an important technology supporting today's emerging data-intensive applications.

FireWire® System Architecture provides an in-depth description of the IEEE 1394a cable environment, based on the 2.0 version of the 1394a standard. Comprehensive, concise, and well-organized, this book details the specification itself and presents the architecture, features, and operations of systems developed with the FireWire® bus. Completely up-to-date, this new edition covers all of the changes incorporated into the 2.0 standard, such as arbitration enhancements, improvements in bus reset timing, new PHY register definition, and port interface enhancements to support suspend/resume and port disable. In addition, it provides in-depth information on suspend/resume operation, power distribution, and power management.

FireWire® System Architecture describes the various hardware and software layers that make up the FireWire® serial bus and provides transaction examples to explain and illustrate the relationship between each layer. It explores such essential topics as:

- The communications model
- Asynchronous and isochronous transactions
- Cables and connectors
- The electrical interface
- Asynchronous and isochronous arbitration, including such enhancements as fly-by arbitration, acceleration control, and priority arbitration service
- Packets, including the PHY packet format

- Serial bus configuration and management
- Control and status registers (CSRs)
- Configuration ROM
- Power management

Numerous sample implementations throughout the book demonstrate how theory is put to use in real-world applications. This combination of comprehensive reference, readable explanation, and examples make this guide indispensable for anyone who works with the IEEE 1394a technology.

The *PC System Architecture Series* is a crisply written and comprehensive set of guides to the most important PC hardware standards. Each title is designed to illustrate the relationship between the software and hardware, and thoroughly explains the architecture, features, and operation of systems built using one particular type of chip or hardware specification.

MindShare, Inc., is one of the leading technical training companies in the computer industry, providing innovative courses for dozens of companies, including Intel, IBM, and Compaq.

Don Anderson, author and co-author of many MindShare books, passes on his wealth of experience in digital electronics and computer design by training engineers, programmers, and technicians for MindShare.

www.awprofessional.com

Cover design by Barbara T. Atkinson
Cover photograph by Tatsuhiko Shimada/Photonica

♻️ Text printed on recycled paper

♣️ ADDISON-WESLEY
Pearson Education



ISBN 0-201-48535-4

\$49.99 US
\$69.99 CANADA