1394

KnowledgeTek

Volume 1

rev 5.41E 15.H.31

Copyright

KnowledgeTek, Inc.

KnowledgeTek specializes in technical training on computer peripherals and related technologies. We offer classes in:

- Disk Drive Technology
- The Cutting Edge
- Disk Drive Servo
- PRML Read Channels
- PRML Lab
- Tape Storage Technology
- Media Noise
- Mag Recording Write Process
- DVD Technology
- The Head/Disk Interface: Advanced Tribology
- SCSI, The Nuts and Bolts
- High Speed SCSI
- IDE, The Nuts and Bolts
- ATAPI
- Fibre Channel
- 1394 Serial Bus
- PCMCIA (PC Card)
- Data Storage Interfaces
- Introduction to SCSI

We can be reached at:

KnowledgeTek, Inc.
7085 West 119th Place
Broomfield, CO 80020
Voice: (303) 465-1800
Fax: (303) 465-2600
email: seminars@KnowledgeTek.com
Web Site: www.knowledgetek.com

Disclaimer

This material is not a specification for any interface or system.

KnowledgeTek makes no warranty and assumes no liability arising out of the application or use of any product, hardware, software, system, circuit, or anything else described herein.

KnowledgeTek assumes no responsibility for errors appearing in this document.

KnowledgeTek assumes no responsibility for any claims that the concepts or details discussed in the seminar, or disclosed in the course materials, are proprietary to any person or company.

Company, brand, product, trade, and other names used in this document are trademarks of their respective holders.

Firewire is a registered trademark of Apple Computer, Inc.

Seminar participants are urged to clear any designs proposed for products with their patent and copyright council.

Trademarks

The 1394 symbol is a registered trademark of Apple Computer, Inc. and used with permission.

About The Instructor

Hugh Curley began working on mainframe computers in 1967 and expanded to personal computers in 1981. His background includes hands-on technical and managerial experience in field service, system-level test in manufacturing, and system-level test in engineering. In 1975 Hugh began teaching computers to engineers and discovered that he not only had good skills for the classroom process, but that he enjoyed teaching working engineers. Now, Hugh has accumulated extensive experience in developing and presenting highly technical courses to engineering specialists from different disciplines. He applies that experience and skill to every course he presents.

Hugh's content strengths are in data storage interfaces, which include IDE and ATAPI, but has particular skill and interest in SCSI. He has successfully presented many interface courses for us.

Current interests have led Hugh to represent KnowledgeTek as an active participant in the 1394A Committee and an interested attendee in other 1394 committees. He is also a member of the IEEE.

Table of Contents

Table of Contents-Appendices

Section 1

1394 Overview

Subjects Covered

Parallel vs Serial

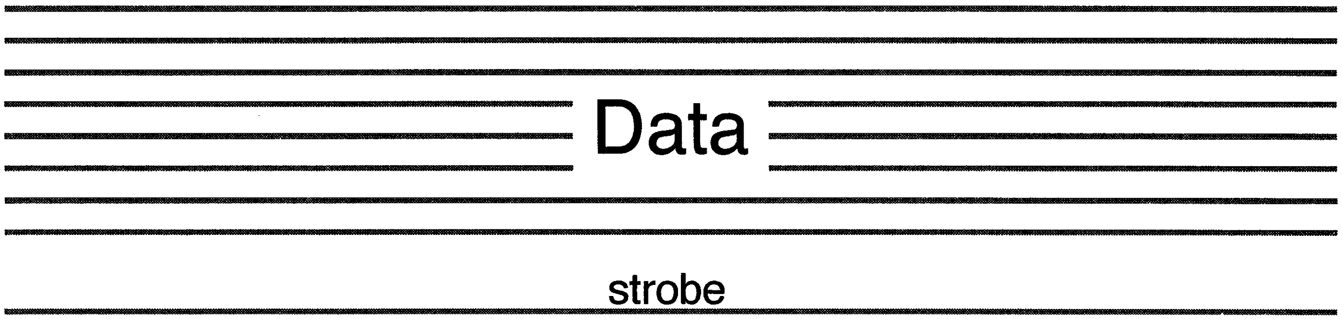
Benefits of 1394

Packets

Isochronous

Other Serial Interfaces

Parallel Interfaces



Termination

Timing Skew

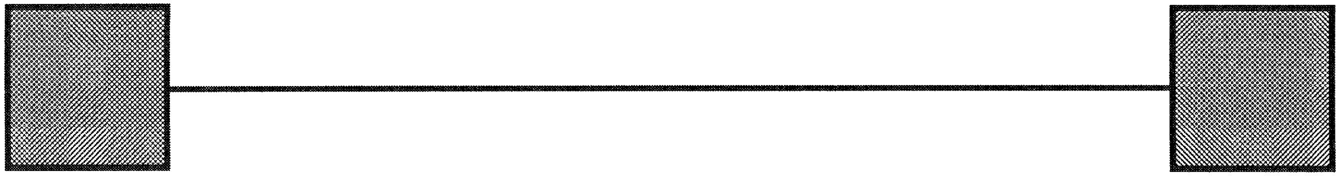
Driver per Bit Wide

Many Signals Change Simultaneously (EMI & Power)

Expensive & Bulky Cables

Expensive & Bulky Connectors

Serial Interfaces



One **Very** Fast Driver rather than Many Fast Drivers

Point to Point Links
- Each Link Terminated
- Flexible Cabling

Great Out-of-Cabinet Solution
- Less RFI
- Cheaper Cables and Connectors

Can Be Made Self-Configuring

1394 Serial Bus

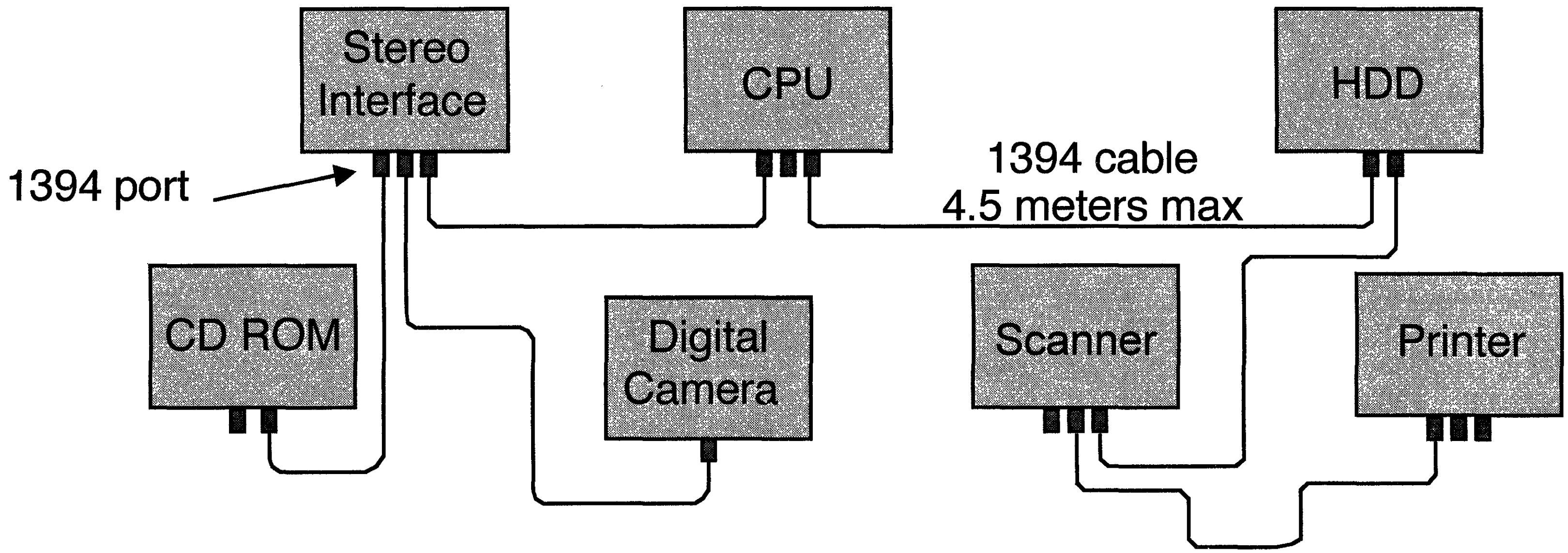
- Targeted at Consumer Market
    - Low Cost
    - Unrestricted Cabling*
    - Supplies Power Over Cable
    - Self-Configuring
    - Multimedia (Scheduled Data Flow)
- Device-Type Independent
- High Speed Data Transfer
- Enabling Protocol for Device Bay

*no loops

1394 Serial Bus Speeds

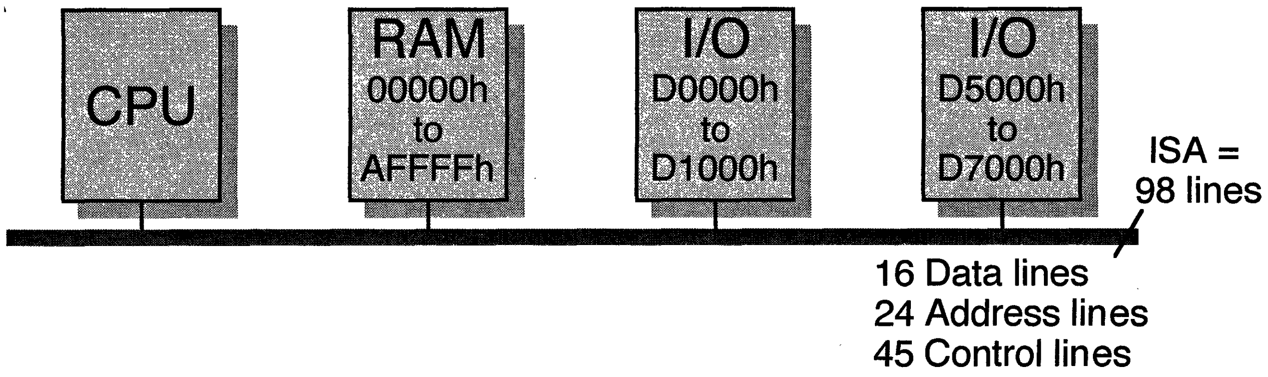
1394 (1995)	S100	100 Mbits/sec (12.5 Mbytes/sec)
	S200	200 Mbits/sec (25 Mbytes/sec)
	S400	400 Mbits/sec (50 Mbytes/sec)
1394b	S800	800 Mbits/sec (100 Mbytes/sec)
	S1600	1600 Mbits/sec (200 Mbytes/sec)

Using 1394 Serial Bus - Physical



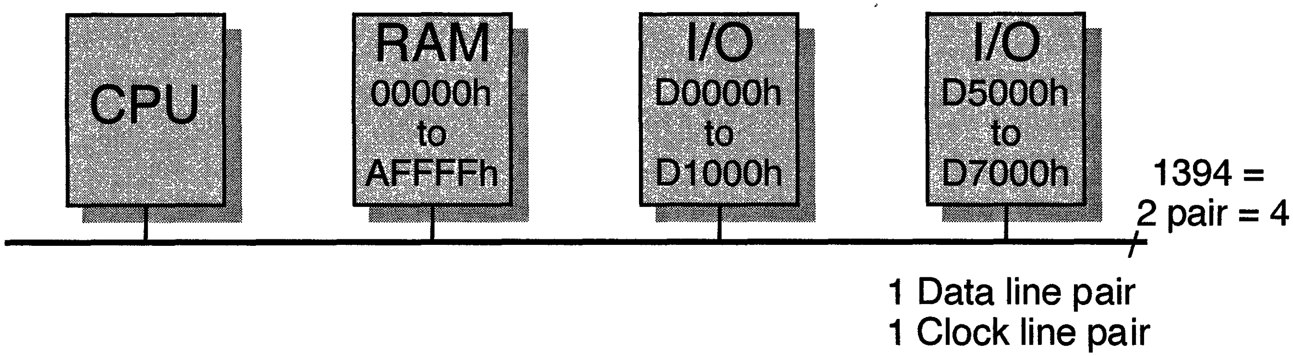
Using 1394 Serial Bus - Logical

Serial Implementation of a Microprocessor Bus



ISA =
98 lines

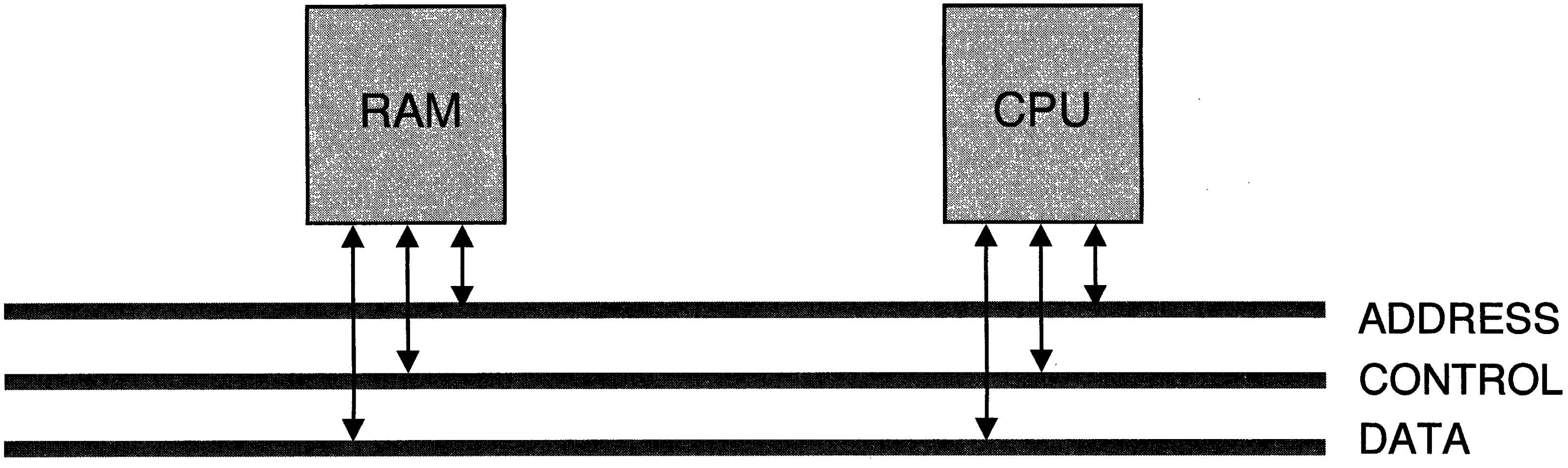
16 Data lines
24 Address lines
45 Control lines

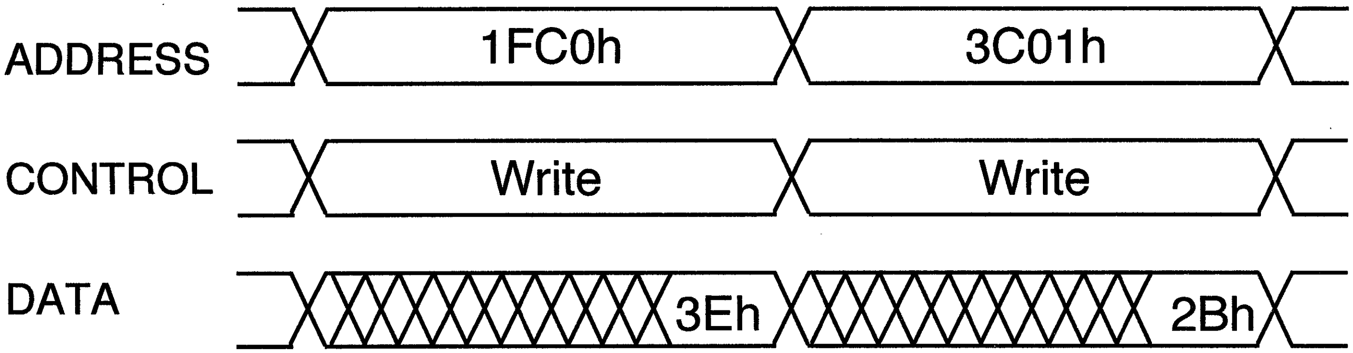


1394 =
2 pair = 4

1 Data line pair
1 Clock line pair

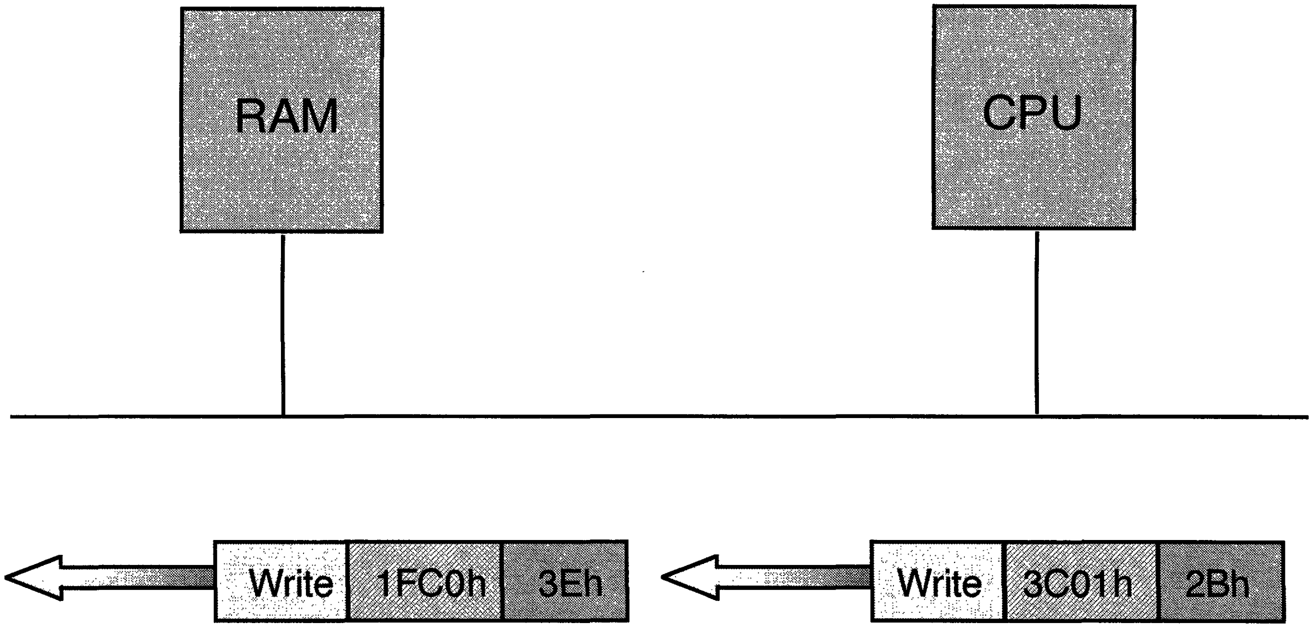
Parallel Microprocessor Bus





Serial Microprocessor Bus

Communication performed with Packet



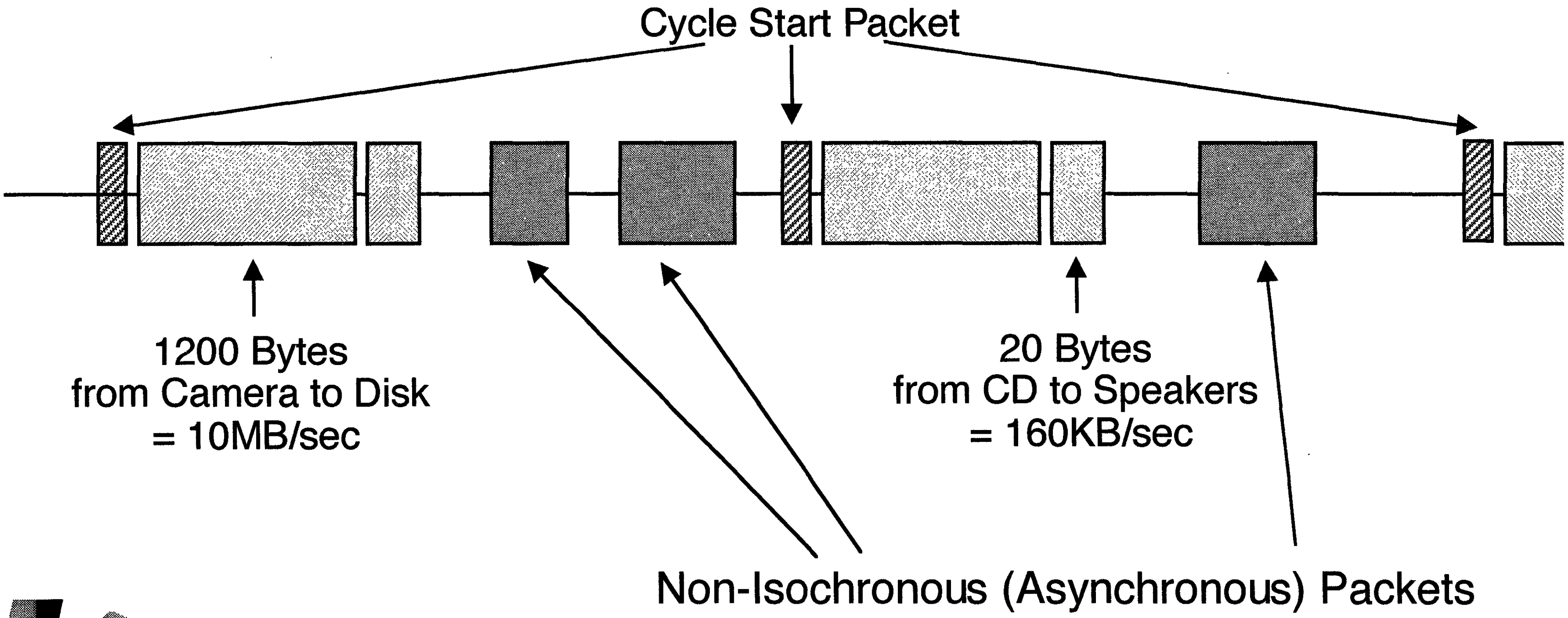
Each Packet contains it's own control and address information

Isochronous

Same Time

Data delivered at a constant rate

Cycle Start packet every 125μSec triggers Isochronous Packets:



Cycle Start Packet

1200 Bytes
from Camera to Disk
= 10MB/sec

20 Bytes
from CD to Speakers
= 160KB/sec

Non-Isochronous (Asynchronous) Packets

Other Serial Interfaces - Fibre Channel

1 & 2 Gbps (200 MByte per second)

Much more expensive per node

Supported Topologies

- Arbitrated Loop
- Fabric (requires switch hardware)
- Point-to-Point

Can go long distances

Designed for High Performance, Cost is Secondary Applications

Other Serial Interfaces - USB

Designed for Lower Speed, Cost is Everything Applications

Keyboard, Mouse, Phone, Printer, etc.

12 Mbps maximum

Not fast enough for Mass Storage or Video

6 meters per segment maximum

Everything controlled directly by PC

USB 2.0 (under development)
will be 30X to 40X faster

Reference - Number System Conversions

Prefix	10^m	2^n
Exa	18	60
Peta	15	50
Tera	12	40
Giga	9	30
Mega	6	20
Kilo	3	10
Hecta	2	
Deca	1	
Unity	0	0
Decia	-1	
Centi	-2	
Milli	-3	-10
Micro	-6	-20
Nano	-9	-30
Pico	-12	-40
Femto	-15	-50
Atto	-18	-60

Decimal	Binary	Hexadecimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Review

1. What are the benefits or target market for 1394?
2. What speeds does 1394 operate?
3. What does a 1394 packet contain?
4. What is Isochronous and how does it operate?

Overview Notes

Section 2

1394 Asynchronous Transactions

Subjects Covered

- Asynchronous Transactions
  - Read
  - Write
  - Lock
- Acknowledges
- Requests and Responses
- Unified and Split Transactions
- Addressing
- Busy Retry

Asynchronous Transactions

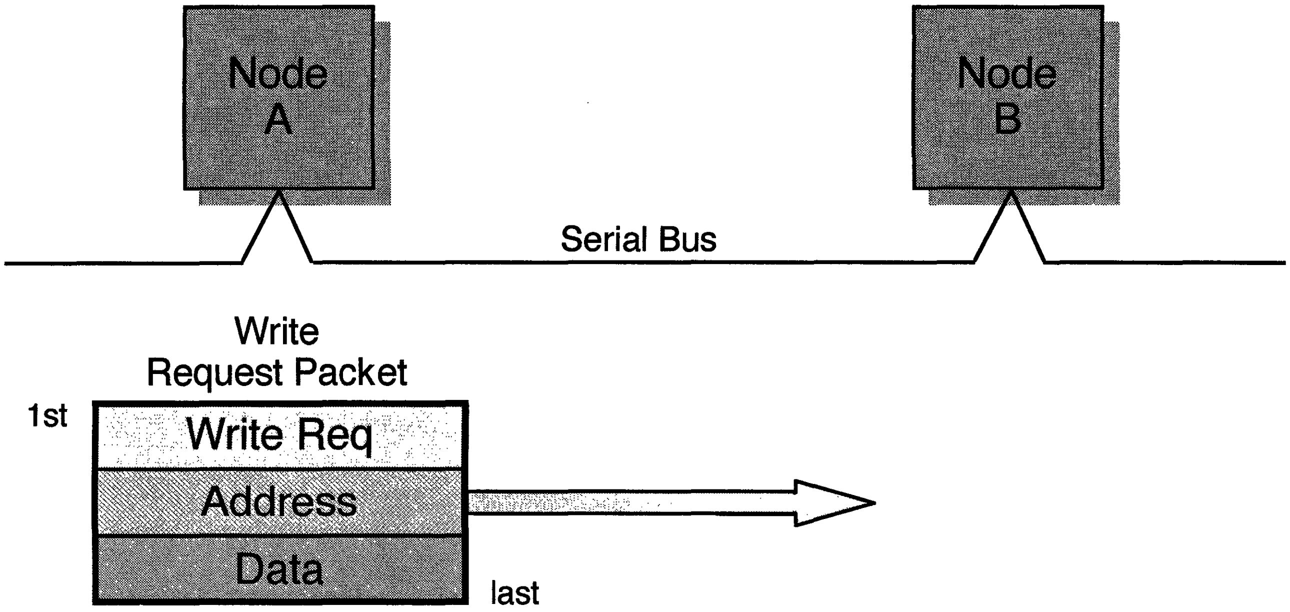
Basic Read and Write Functions

Not synchronized to time

Normally does not include audio, video, multimedia

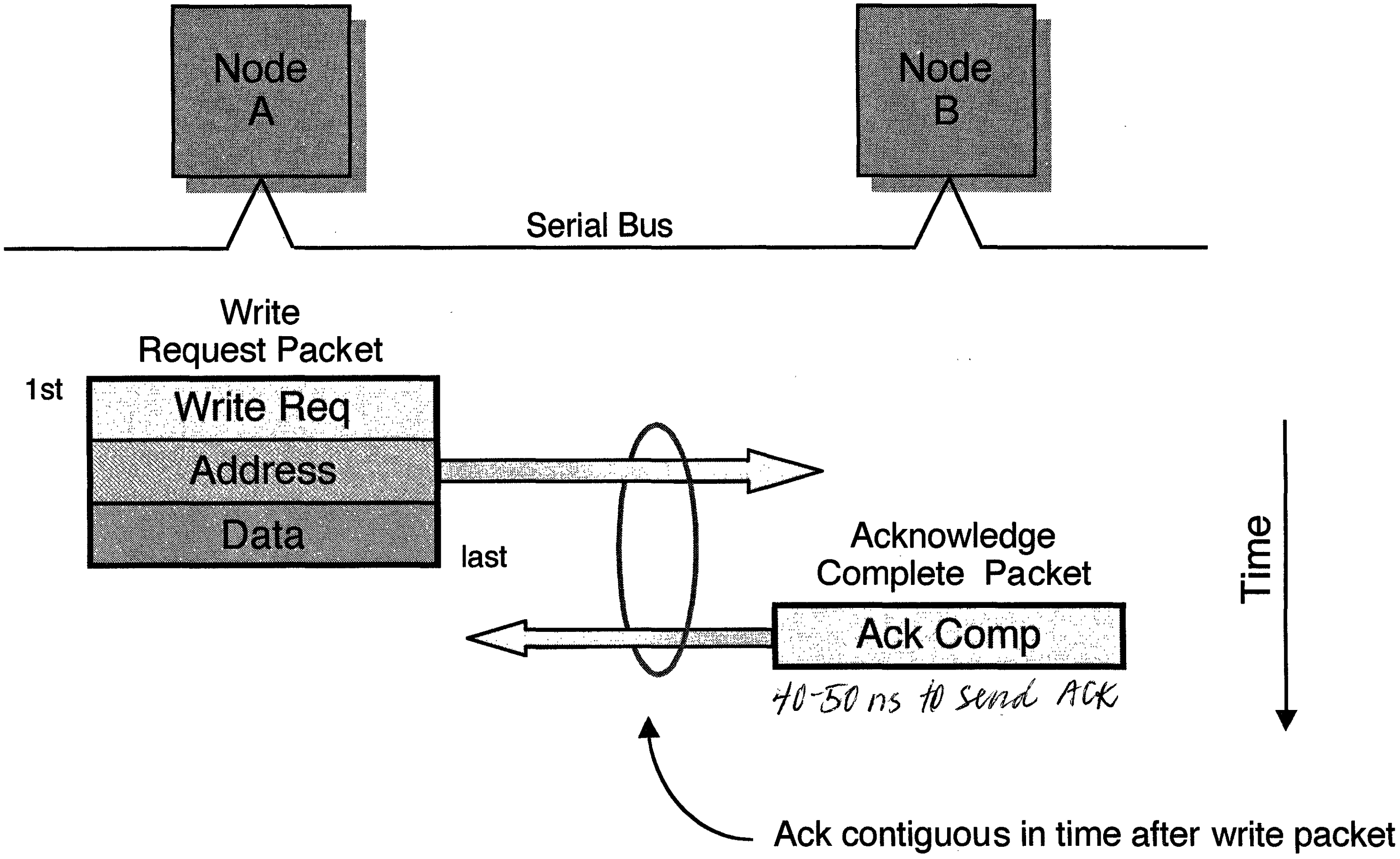
Accuracy of data delivery is more critical than timing

Write Transaction



How does node A know if the packet is received correctly?

Complete Write Transaction



Ack contiguous in time after write packet

Acknowledges

Contiguous on the bus following the write data

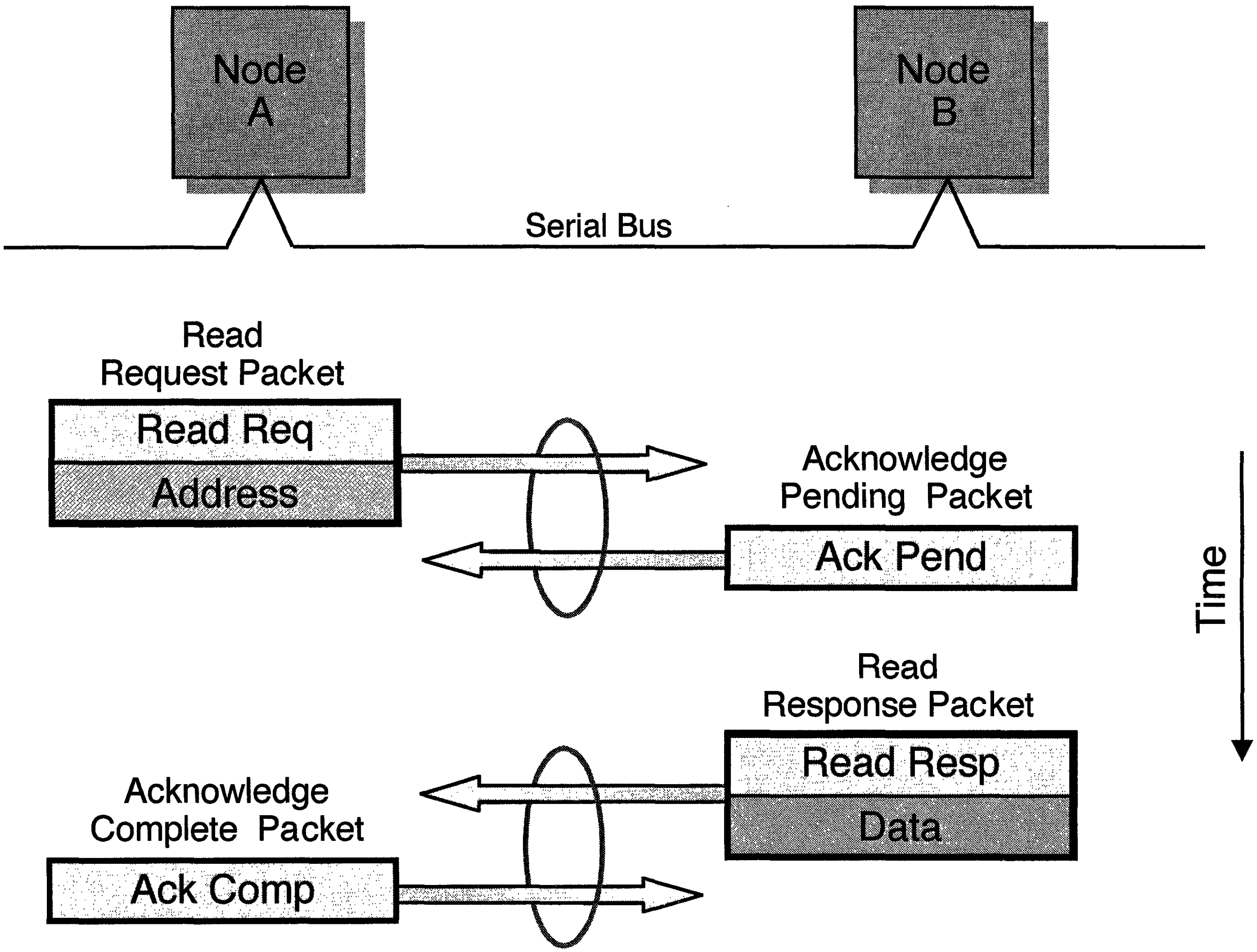
Several Types:

- Complete — Operation completed satisfactorily
- Busy (A/B/X) — Operation not completed, receiver node busy
- Error (data/type) (address/conflict) — Operation not completed, there was an error
- Pending — Operation not completed, but is being processed
- Tardy — Node will take a while to respond (in low power state)

Each type will be described in detail later

What about a read that requires a seek or search, and the data is delayed?

Read Operation



Node A

Node B

Serial Bus

Read Request Packet

Read Req

Address

Acknowledge Pending Packet

Ack Pend

Time

Read Response Packet

Read Resp

Data

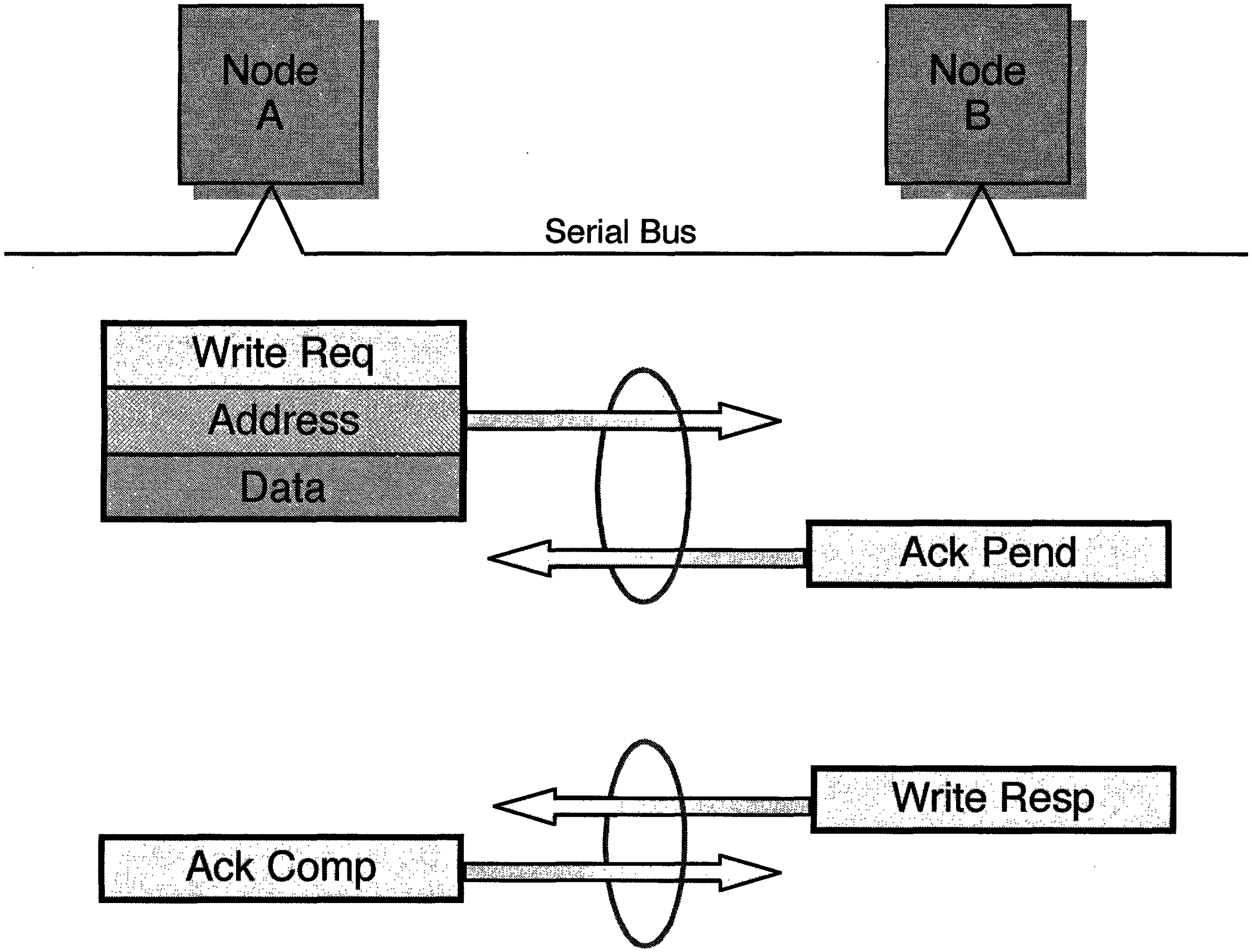
Acknowledge Complete Packet

Ack Comp

This is referred to as **Split Response**

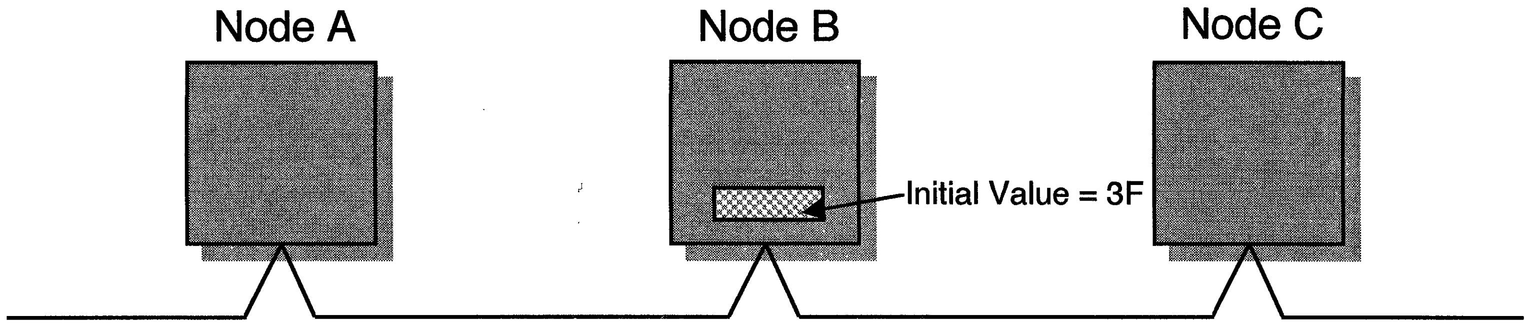
all Reads are ↑

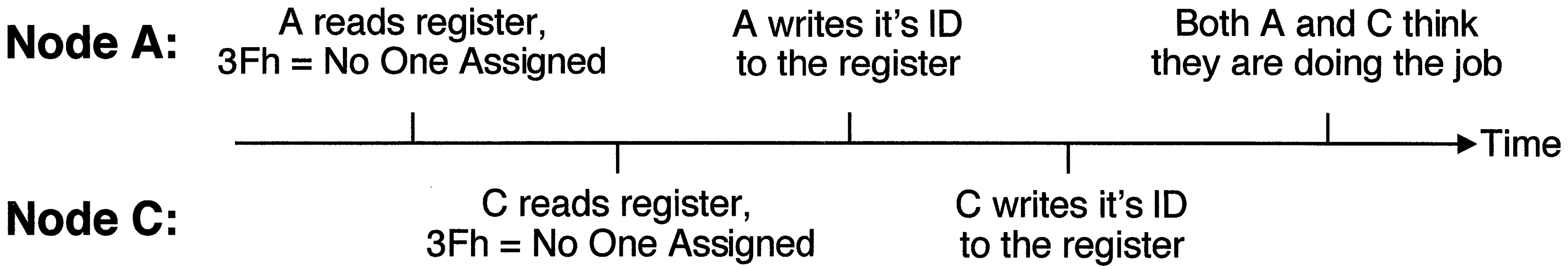
Non-Unified Write Operation
Split Response



Coherence Problem

A register on Node B indicates who is going to perform a given function
The register is initialized to 3Fh indicating no one has been assigned
Both A and C want to perform this function:





Node A:

A reads register,
3Fh = No One Assigned

A writes it's ID
to the register

Both A and C think
they are doing the job

Node C:

C reads register,
3Fh = No One Assigned

C writes it's ID
to the register

Coherence Solution - Lock Transaction

Makes testing a flag and setting it one action

Required because of split response nature of 1394

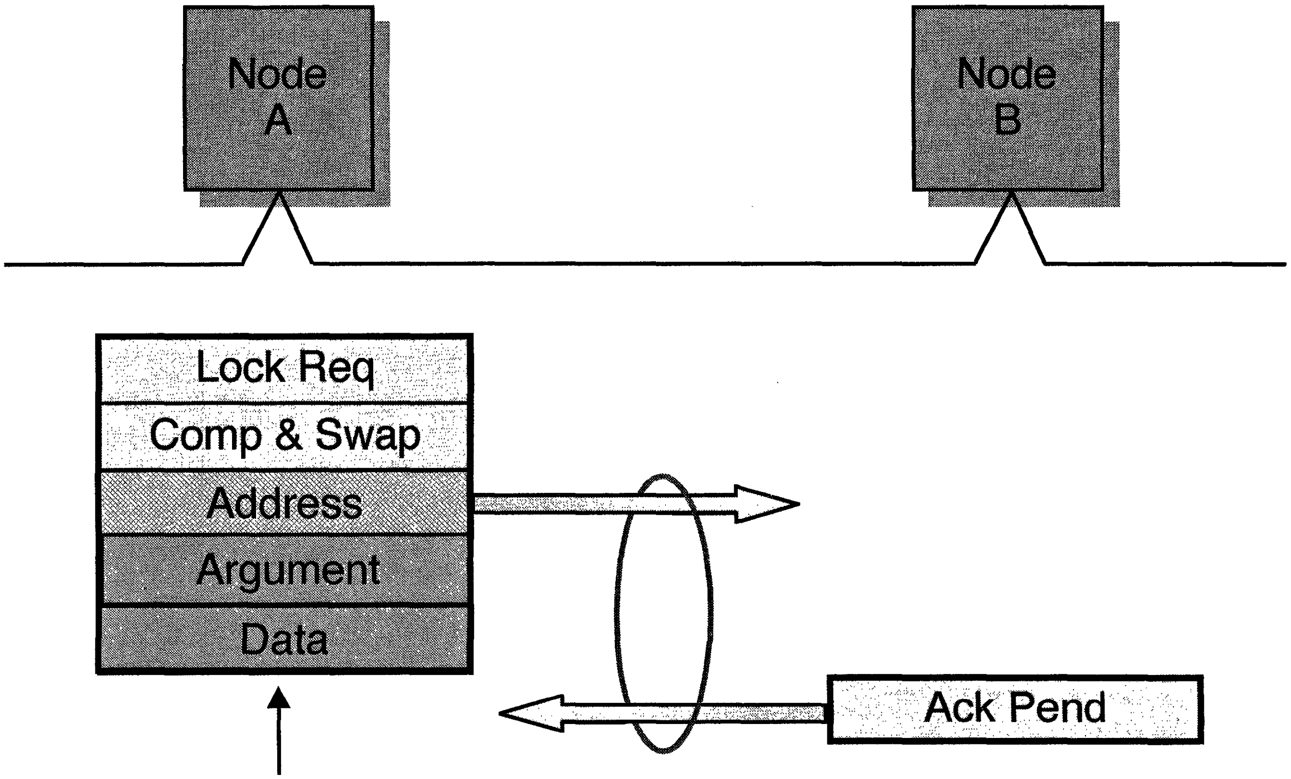
Basic Functions:

- **Compare and Swap**
- Mask and Swap
- Fetch and Add
- Little Add
- Bounded add
- Wrap Add

Not used by 1394

Used to communicate with some CSRs (section 3)

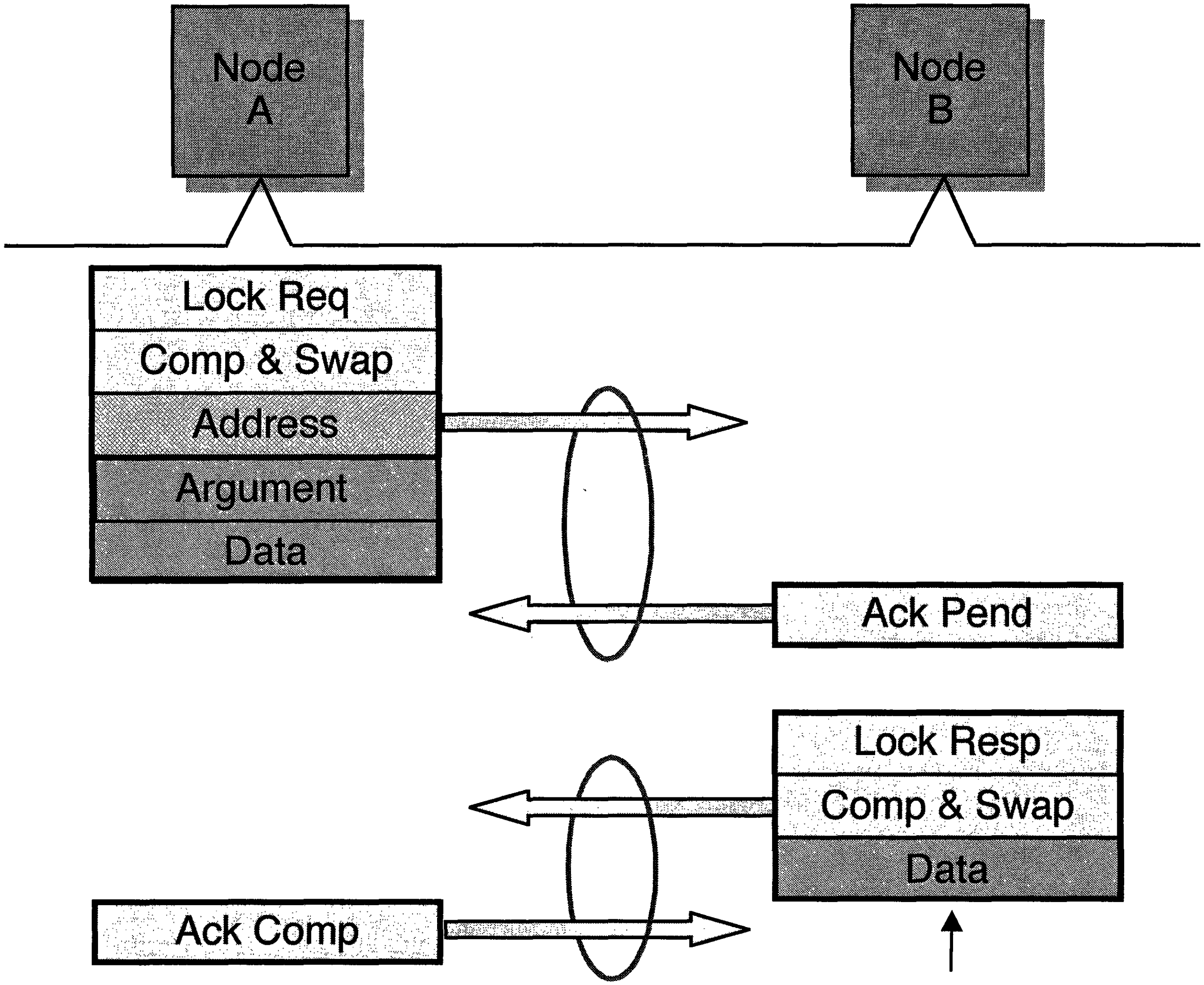
Lock Transaction



Asks Node B to:

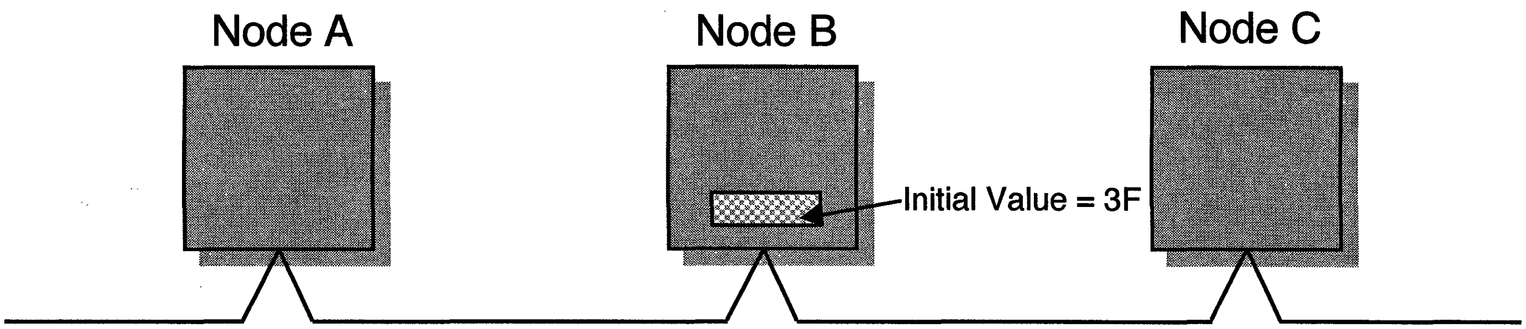
- Check location Address
- Compare to Argument
- If the same, replace with Data

Lock Response



Data before
Compare and Swap

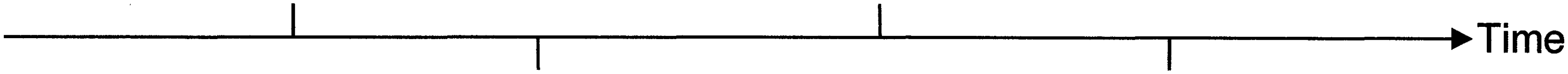
Solving Coherence Problems With Lock Transactions



Node A:

Lock Req to B
Addr = Register Loc
Arg = 3F
Data = Node A ID

Lock Resp to A
Data = 3F
Indicates previously unassigned
Node A performs the function



Node C:

Lock Req to B
Addr = Register Loc
Arg = 3F
Data = Node C ID

Lock Resp to C
Data = Node A ID
Indicates Node A performs the function

For More Information: Other Lock Functions

Mask & Swap	Set all bits that are '1' in data Clear all bits that are '0' in argument
Fetch & Add	Add data
Bounded Add	If memory ≠ argument, add data
Add Little	Fetch & Add but in little-endian order
Wrap & Add	If memory ≠ argument, add data otherwise set to data

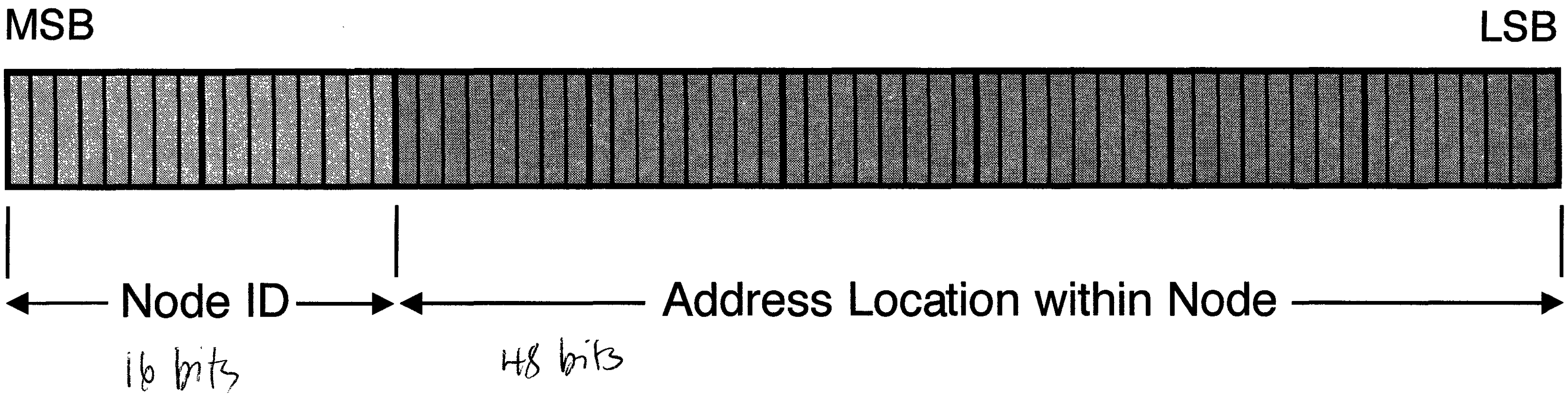
For More Information: Lock Functions in C

```
Compare & Swap      if (old_value == arg_value)
                        new_value = data_value;
                    else new value = old_value;

Mask & Swap         new_value = data_value | (old_value & ~arg_value);

Fetch & Add         new_value = old_value + data_value;

Bounded Add         if (old_value != arg_value)
                        new_value = old_value + data value;
                    else new value = old_value;

Little Add          new_value = LittleEndAdd (old_value, data_value);

Wrap & Add          if (old_value != arg_value)
                        new_value = old_value + data_value;
                    else new_value = data_value;
```

Addressing

64 bit Addresses

2^{64} = 16 ExaBytes Addressed



MSB

LSB

Node ID

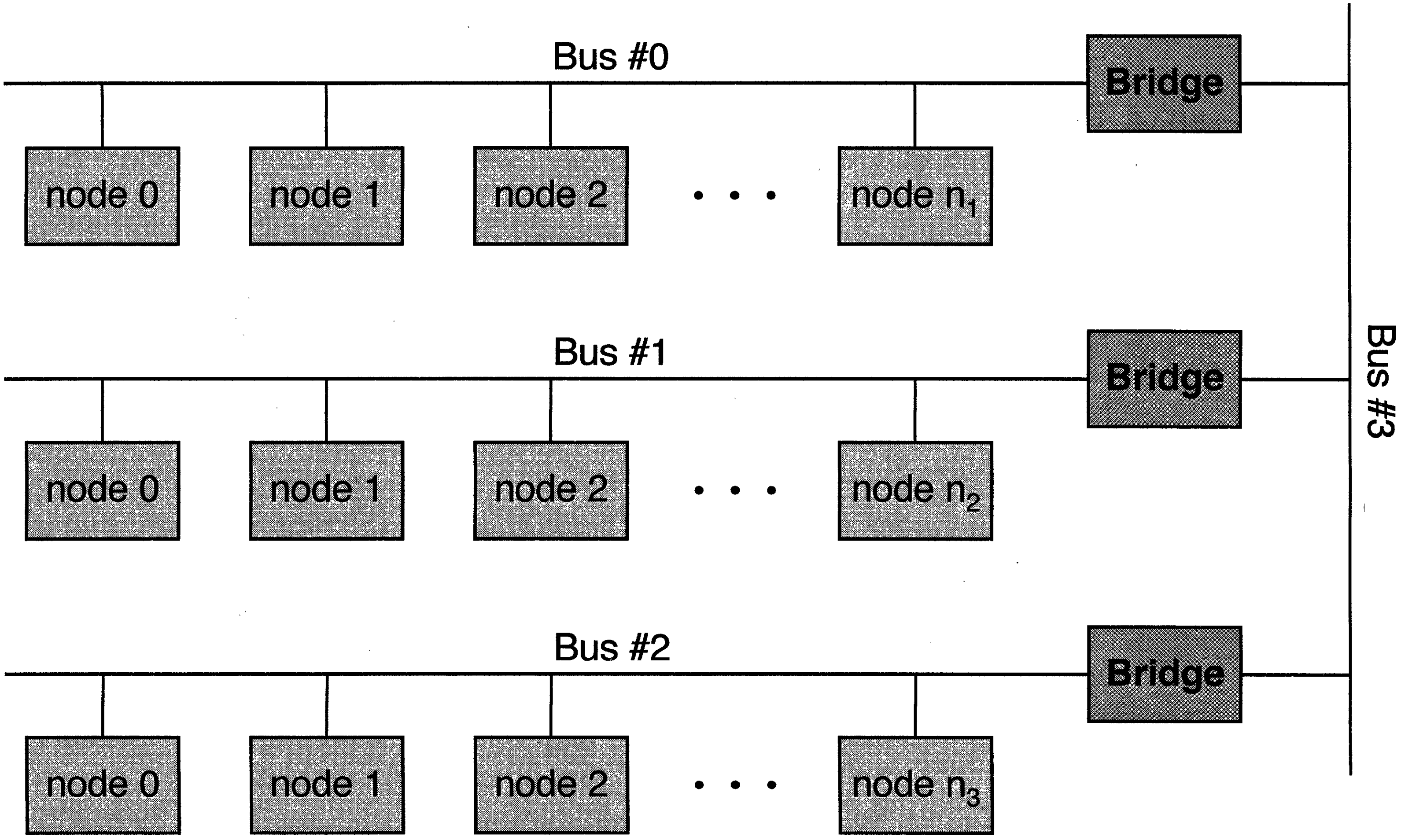
Address Location within Node

16 bits

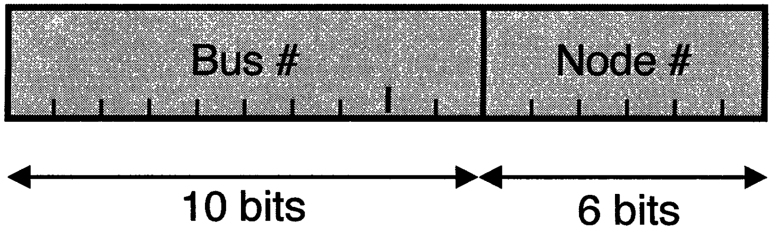
48 bits

Each Node has 2^{48} Bytes = 256 TeraBytes of Address Range

Nodes And Busses



Node ID



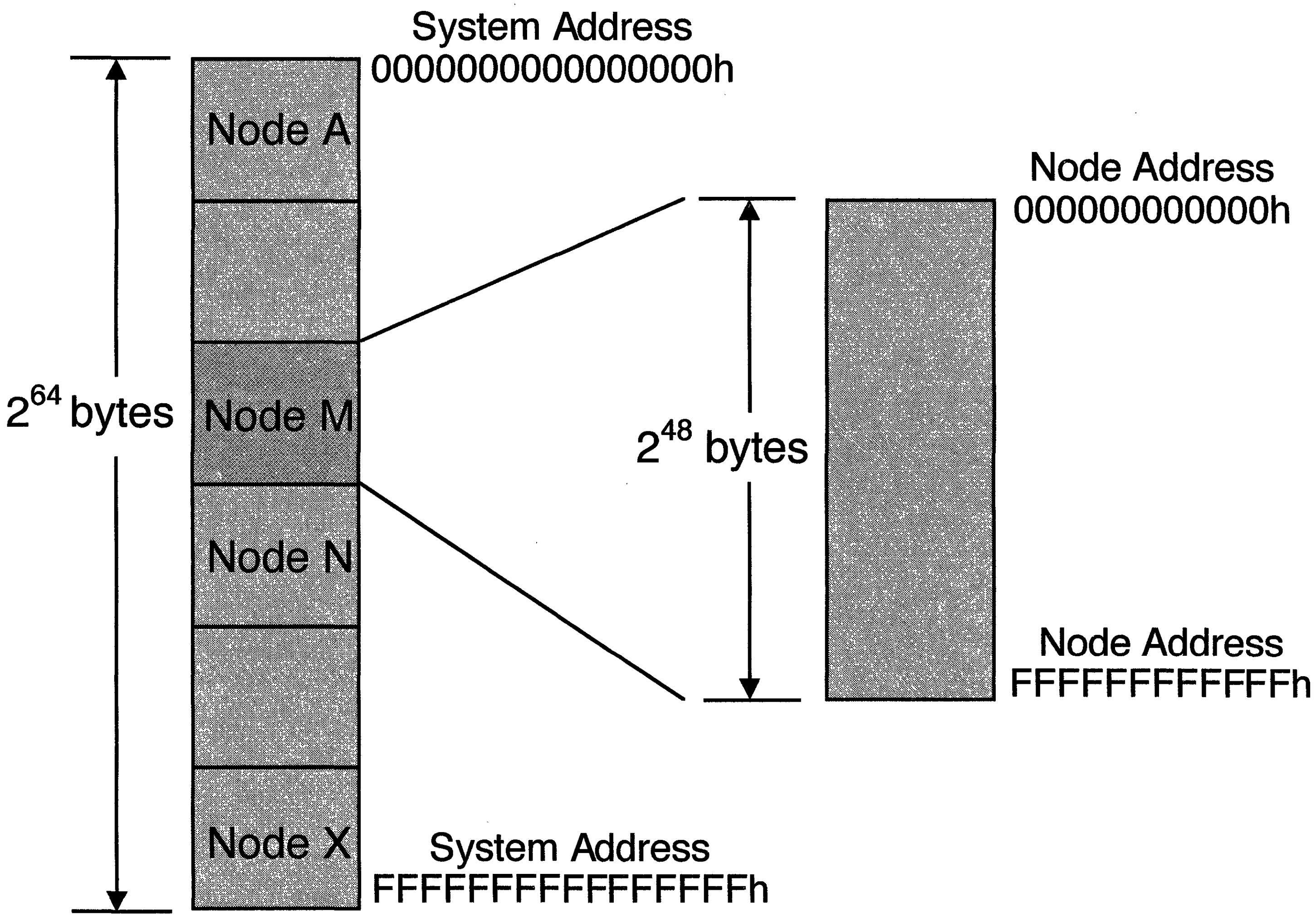
1023 Busses maximum (0 - 1022)

63 Nodes on a Bus maximum (0 - 62)

Bus 3FFh is local bus

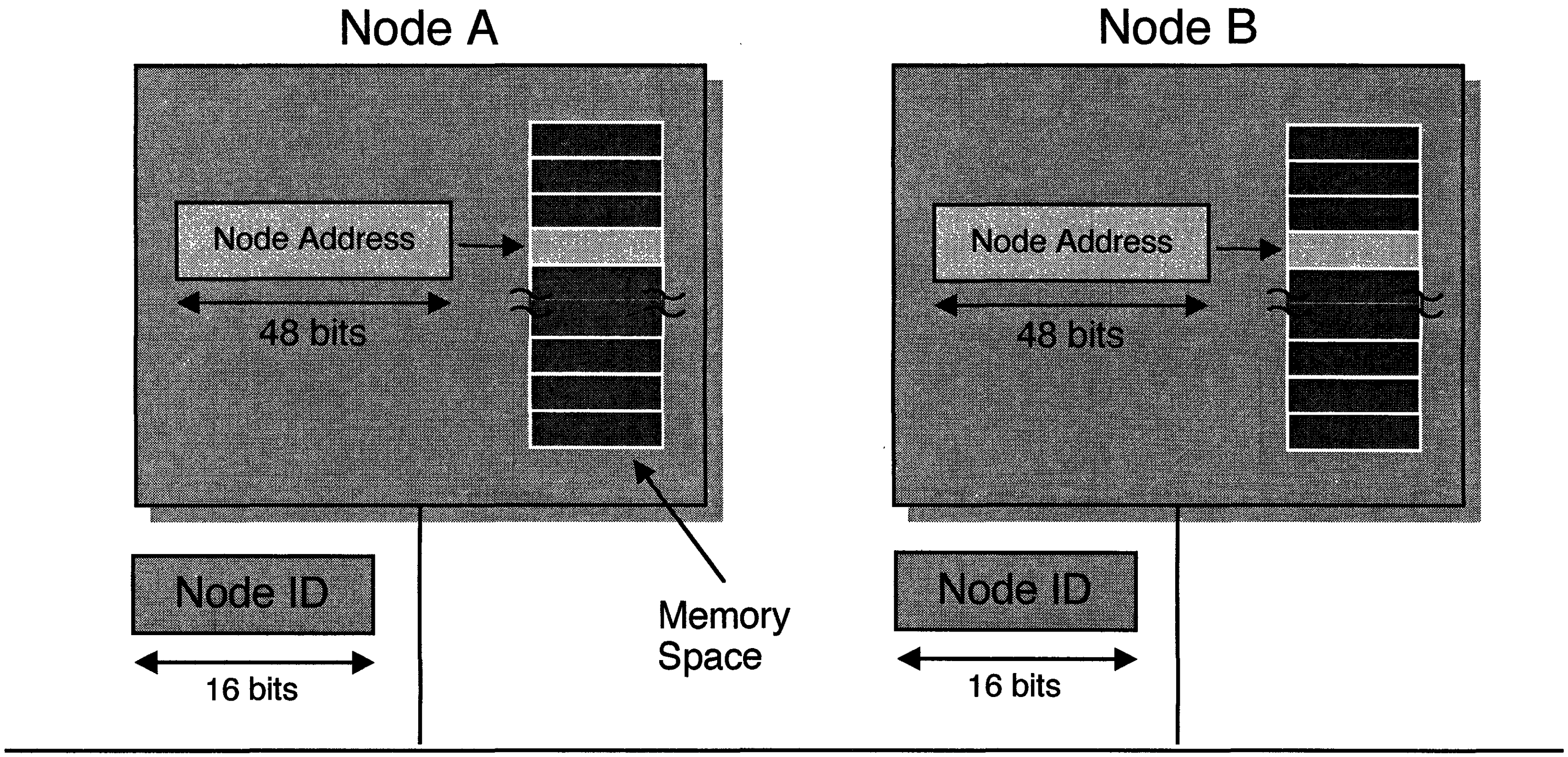
Node 3Fh broadcasts to all nodes on indicated bus

1394 Address Map

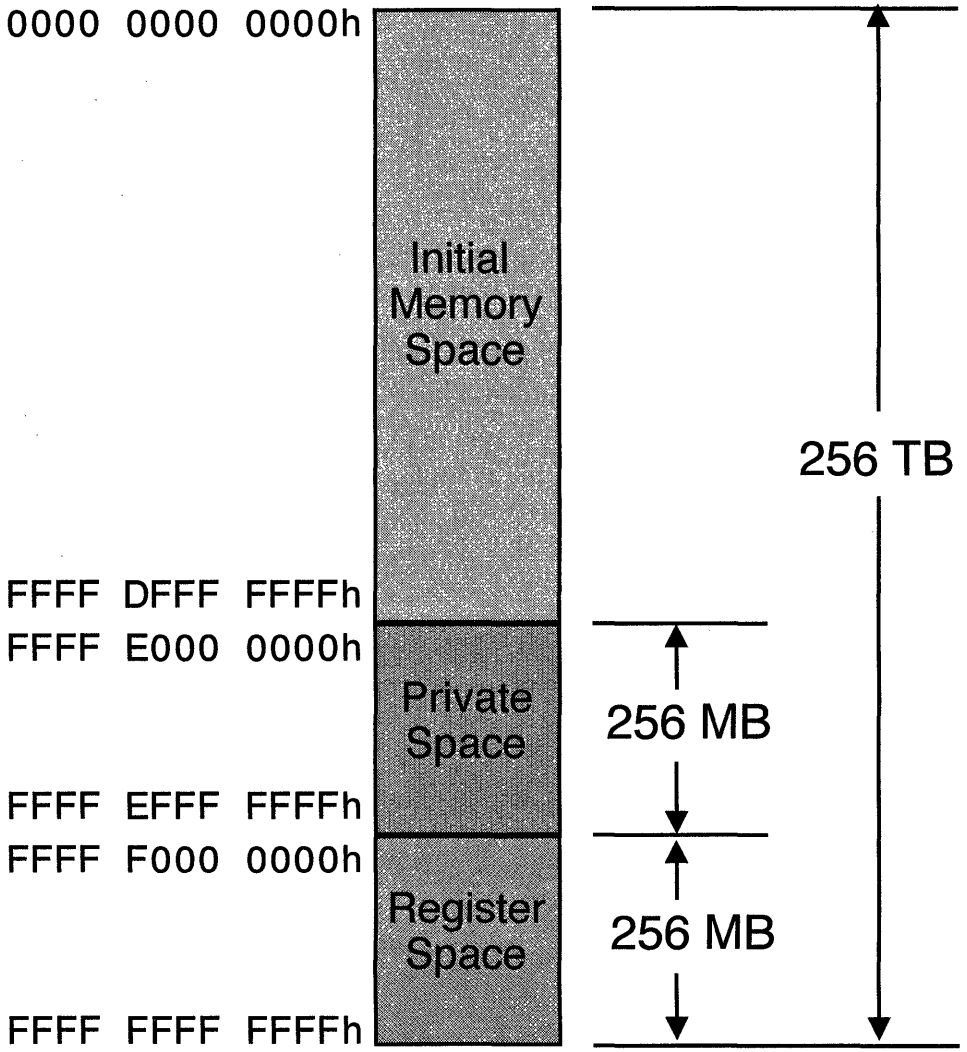


Addresses are shown from top - down throughout this course

Addressing - Different view



Node Memory Space



1394 does not use private space

Size Notations

Size in bits	16 bit word machine notation	32 bit word machine notation	IEEE standard notation
8	Byte	Byte	**Byte**
16	Word	Half-word	Doublet
32	Long-word	Word	**Quadlet**
64	Quad-word	Double Word	Octlet

Octet

Used in this course

Byte Ordering

transmitted first

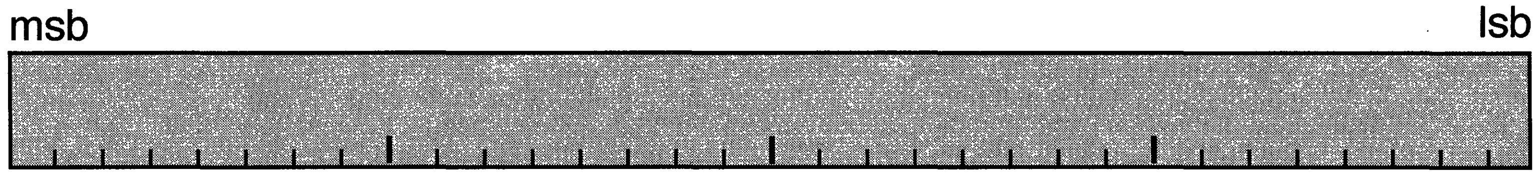
Quadlet 0	msb Byte 0	Byte 1	Byte 2	lsb Byte 3
Quadlet 1	Byte 4	Byte 5	Byte 6	Byte 7
Quadlet 2	Byte 8	Byte 9	Byte 10	Byte 11
	...	...	...	...
Quadlet m-1	Byte n-7	Byte n-6	Byte n-5	Byte n-4
Quadlet m	Byte n-3	Byte n-2	Byte n-1	Byte n

transmitted last

$$m = \frac{n-3}{4}$$

Bit and Byte Ordering

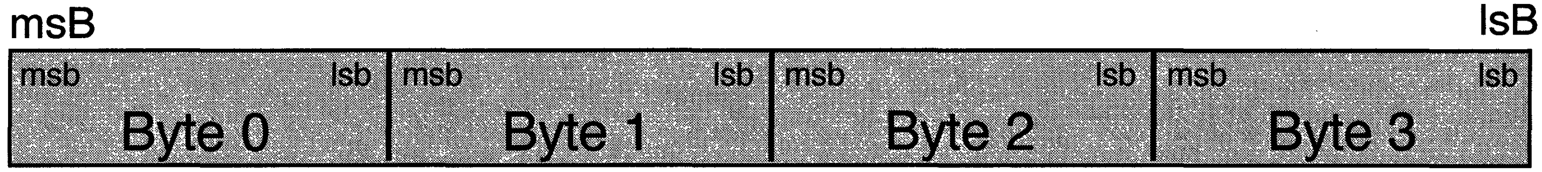
Bit Ordering



msb

lsb

Bytes in a Quadlet



msB

lsB

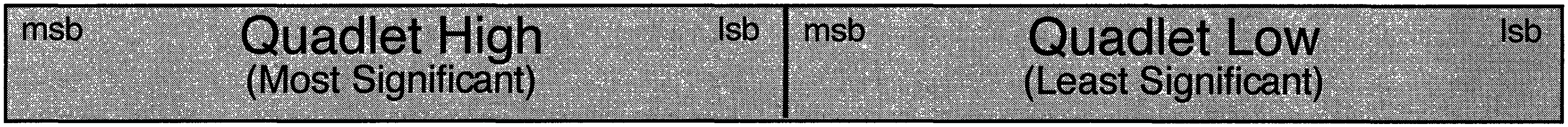
msb lsb Byte 0

msb lsb Byte 1

msb lsb Byte 2

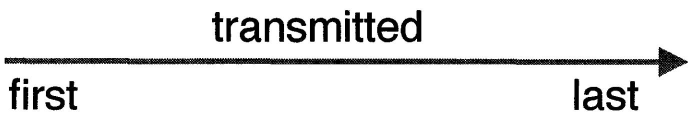
msb lsb Byte 3

Quadlets in an Octlet



msb Quadlet High (Most Significant) lsb

msb Quadlet Low (Least Significant) lsb



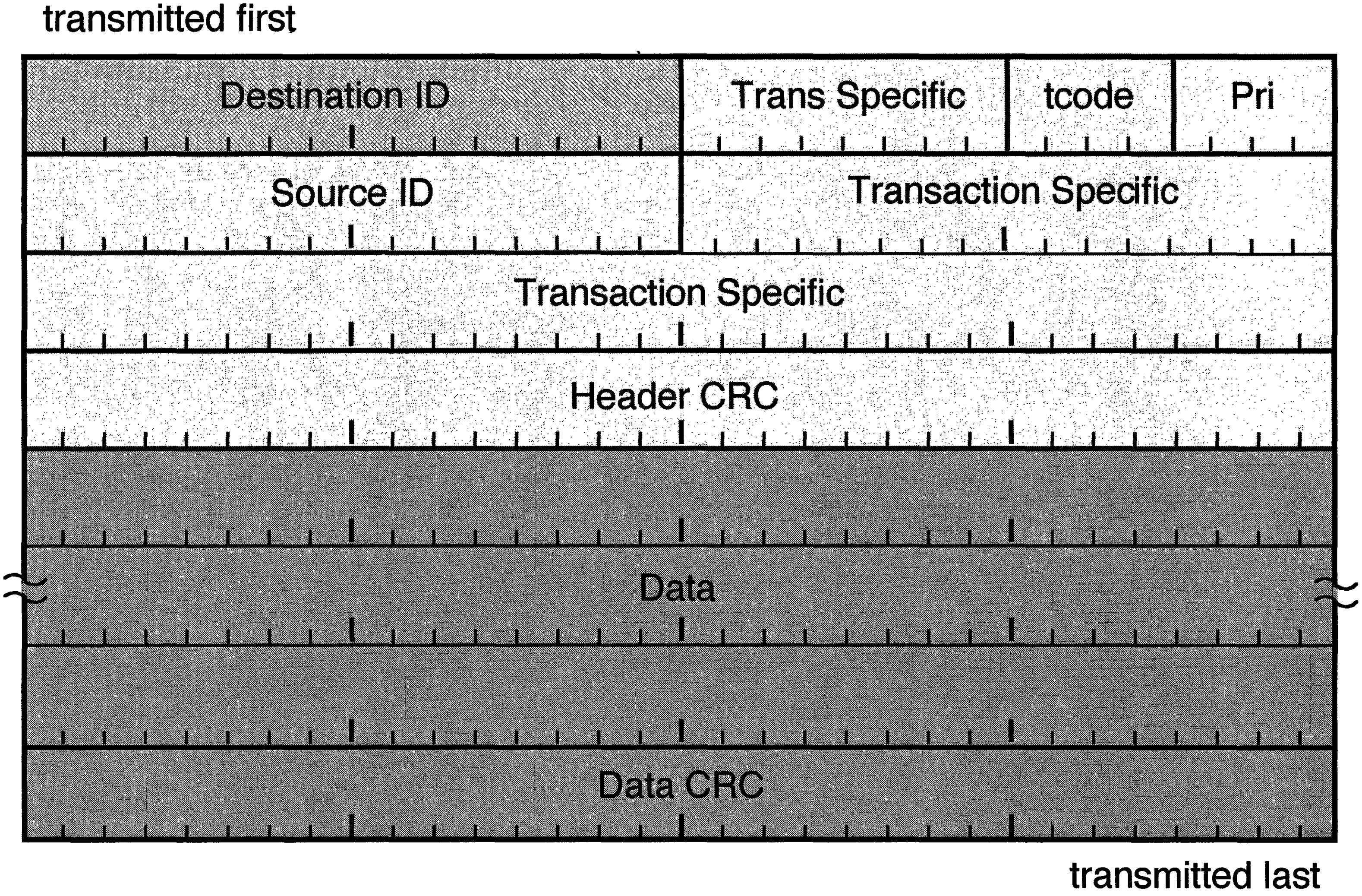
transmitted

first

last

this page is intentionally blank

Generic Packet Format



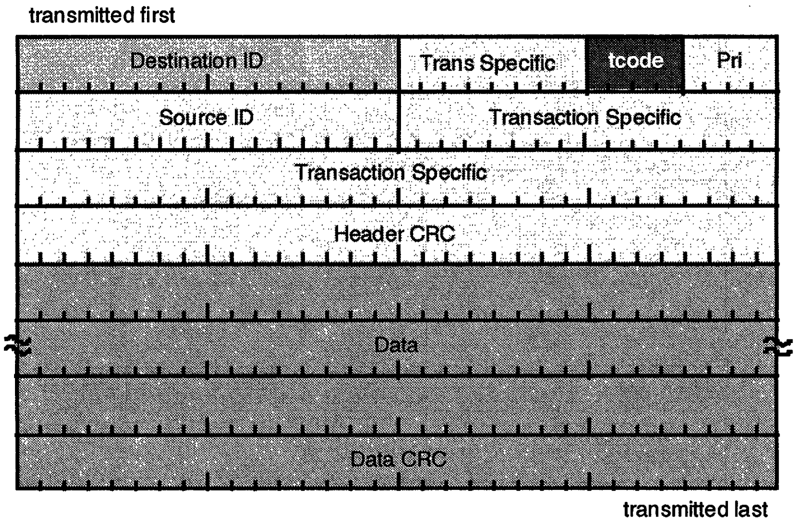
Generic Packet Format Definitions

Destination ID	High order 16 bits of address designating receiving node.
Source ID	High order 16 bits of sending node
tcode	Transaction code, identifies this as read, write, etc.
Pri	Priority, meaningful on backplane implementations only
Header CRC	32 bit Cyclic Redundancy Check for header quadlets.
Data CRC	32 bit Cyclic Redundancy Check for data quadlets.

V

CRC is the same CRC used by IEEE 802 and FDDI.

Transaction Codes (tcode)

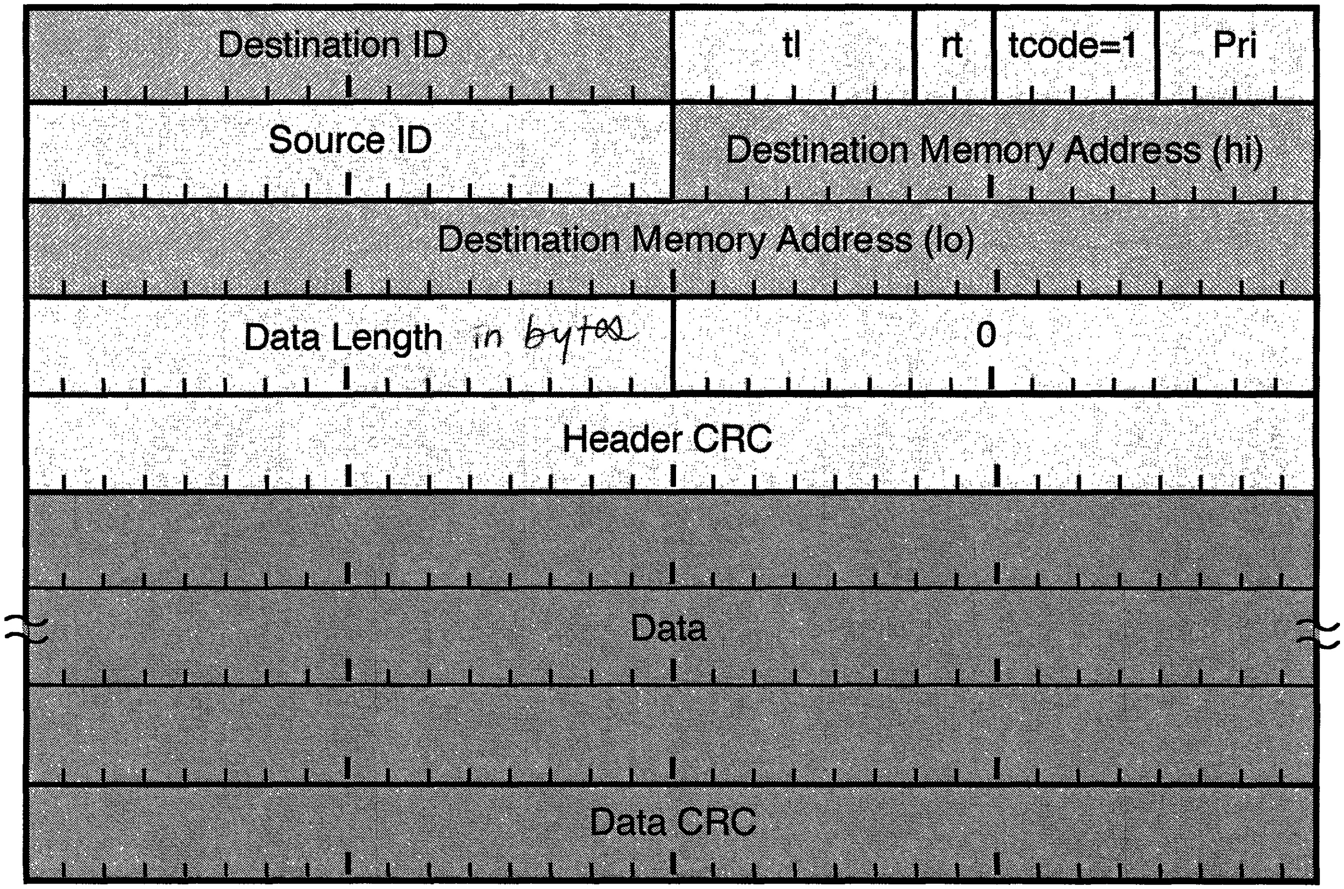


tcode		function
0000	0h	Write Req for quadlet
0001	1h	Write Req for block
0010	2h	Write Response
0011	3h	Reserved
0100	4h	Read Req for quadlet
0101	5h	Read Req for block
0110	6h	Read Response for quadlet
0111	7h	Read Response for block
1000	8h	Cycle Start
1001	9h	Lock Request
1010	Ah	Stream Packet
1011	Bh	Lock Response
1100	Ch	Reserved
1101	Dh	Reserved
1110	Eh	Reserved
1111	Fh	Reserved

this page is intentionally blank

Block Write Request Packet Format

transmitted first



Destination ID	tl	rt	tcode=1	Pri
Source ID	Destination Memory Address (hi)			
Destination Memory Address (lo)				
Data Length in bytes	0			
Header CRC				
Data				
Data CRC				

transmitted last

pad to quadlet length

Pad zero bytes on end of data block if needed

Write Request Packet Format

Destination ID	16 bit ID of receiving node
Source ID	16 bit ID of sending node
tl	Transaction Label - defined later
rt	Retry Code - defined later
tcode = 1	Transaction code, identifies this packet as block write request
Pri	Priority, only meaningful on backplane
CRC	Check data for header or data (including pad bytes)
Data Length	Number of bytes in data field (does not include pad bytes)

V

Block Read Request Packet Format

transmitted first

<table>
<tr><td colspan="8">Destination ID who?</td><td colspan="3">tl</td><td colspan="1">rt</td><td colspan="2">tcode=5</td><td colspan="2">Pri</td></tr>
<tr><td colspan="8">Source ID</td><td colspan="8">Destination Memory Address (hi)</td></tr>
<tr><td colspan="16">Destination Memory Address (lo) where?</td></tr>
<tr><td colspan="8">Data Length bytes</td><td colspan="8">0</td></tr>
<tr><td colspan="16">Header CRC</td></tr>
</table>

transmitted last

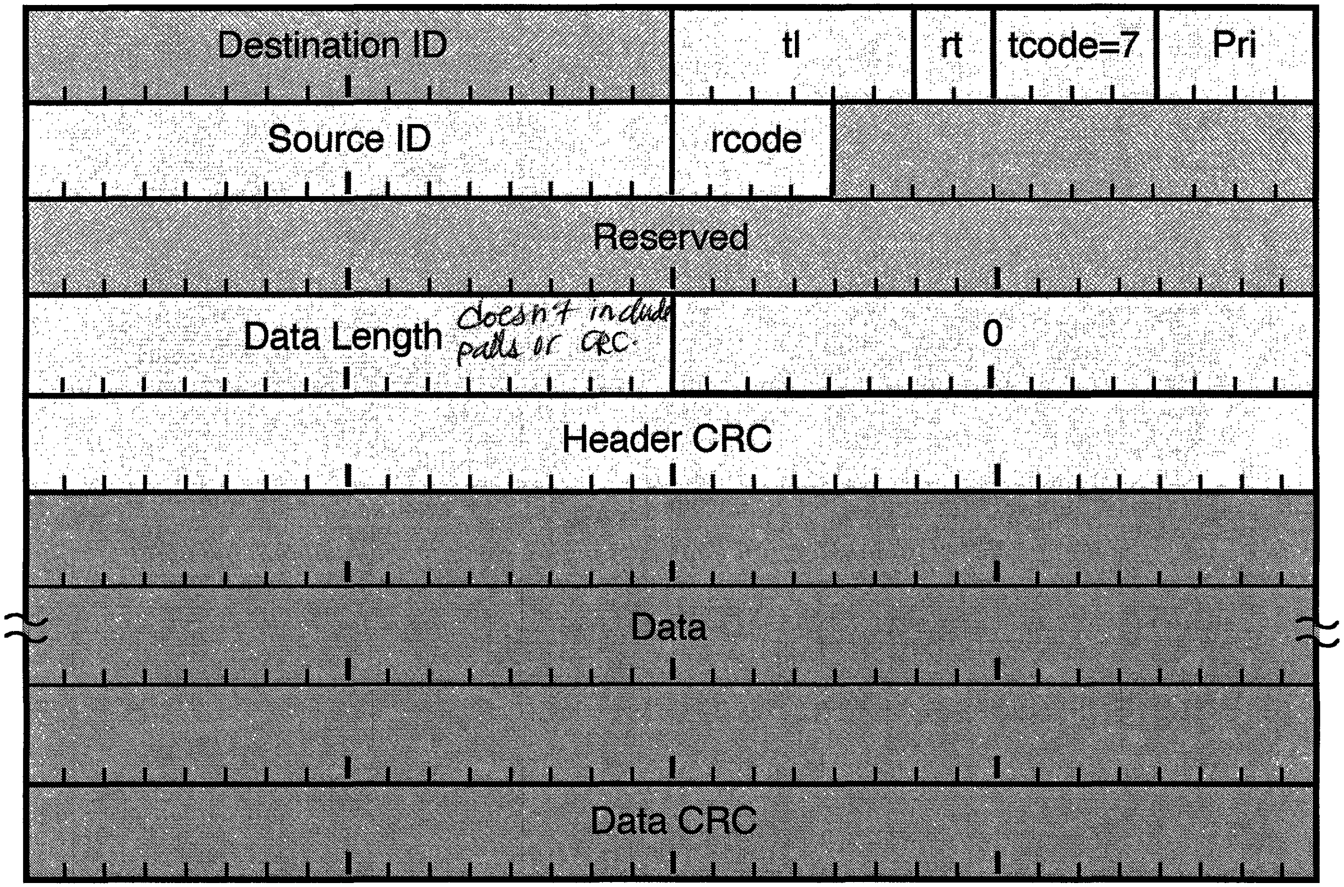
Read Request for Data Block Packet Definitions

Destination ID	Node ID of receiving node
Source ID	Node ID of requesting node
tl	Transaction label
rt	Retry code
tcode = 5	Transaction code, 5 = Read Block Request
Pri	Priority, for backplane environment
Memory address	48 bit address within the node. This concatenated with the Destination ID is the 64 bit system address.
Data Length	Length of expected data in bytes

V

Block Read Response Packet Format

transmitted first



Destination ID | tl | rt | tcode=7 | Pri

Source ID | rcode

Reserved

Data Length doesn't include pads or CRC. | 0

Header CRC

Data

Data CRC

transmitted last

Pad zero bytes on end of data block if needed

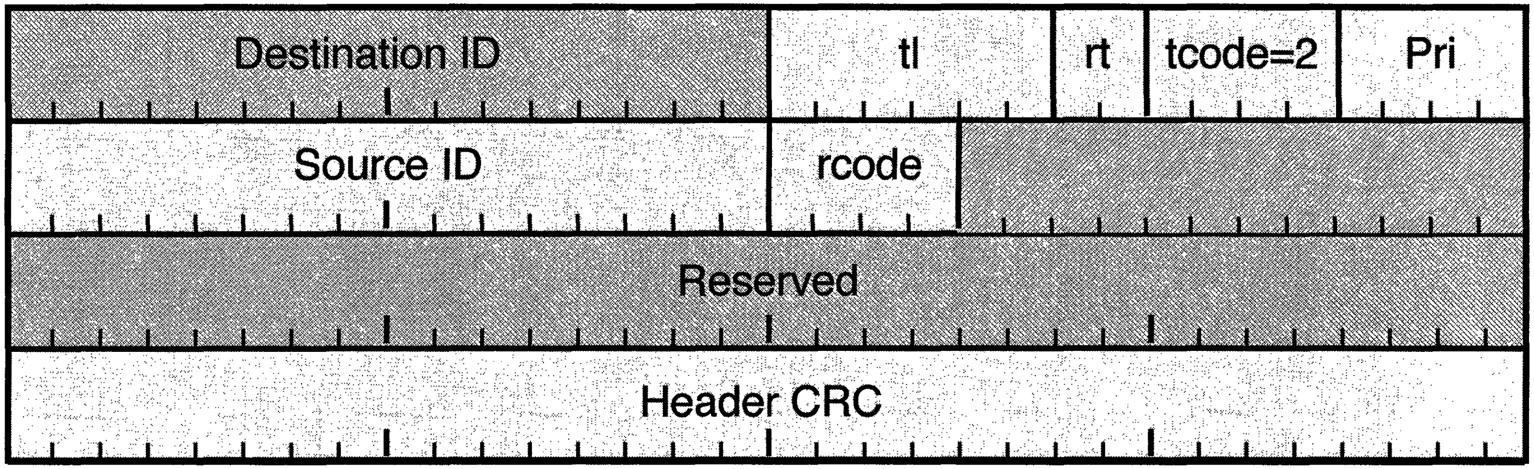
Read Response for Data Block Packet Definitions

Destination ID	Node ID of receiving node
Source ID	Node ID of requesting node
tl	Transaction label
rt	Retry code
tcode = 7	Transaction code, 7 = Read Block Response
Pri	Priority, for backplane environment
rcode	Response Code - described later
Data Length	Length of data
Header CRC	32 bit CRC check for header information
Data field	Data payload
Data CRC	32 bit CRC check for data field

V

Write Response Packet Format

transmitted first



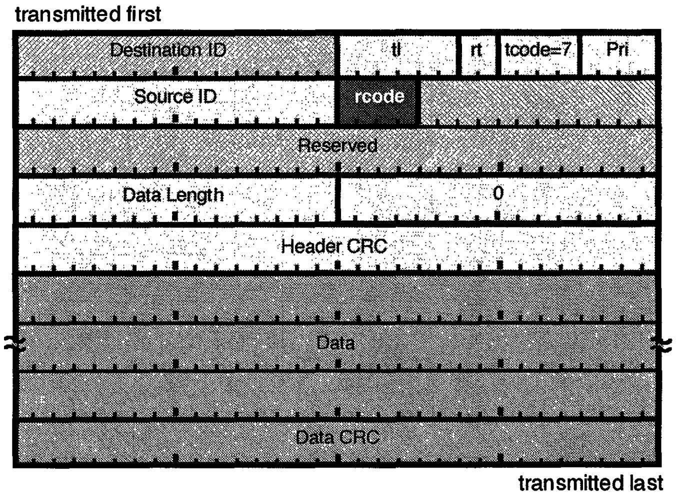
transmitted last

Write Response Packet Format Definitions

Destination ID	16 bit ID of receiving node
Source ID	16 bit ID of sending node
tl	Transaction Label
rt	Retry Code
tcode = 2	Transaction code, 2 = write response
Pri	Priority, only meaningful on backplane
rcode	Response Code
Header CRC	32 bit Cyclic Redundancy Check for header quadlets.

V

Response Codes (rcode)



rcode		meaning
0000	0h	Transaction completed successfully
0001	1h	Reserved
0010	2h	Reserved
0011	3h	Reserved
0100	4h	Resource conflict (retry)
0101	5h	Hardware data error (data not available)
0110	6h	Illegal request (invalid operation or unsupported value)
0111	7h	Unavailable Address
1000 to 1111	8h to Fh	Reserved

FFC7

1111 1111 11|00 0007

local bus | node7

Exercise: What's Happening?

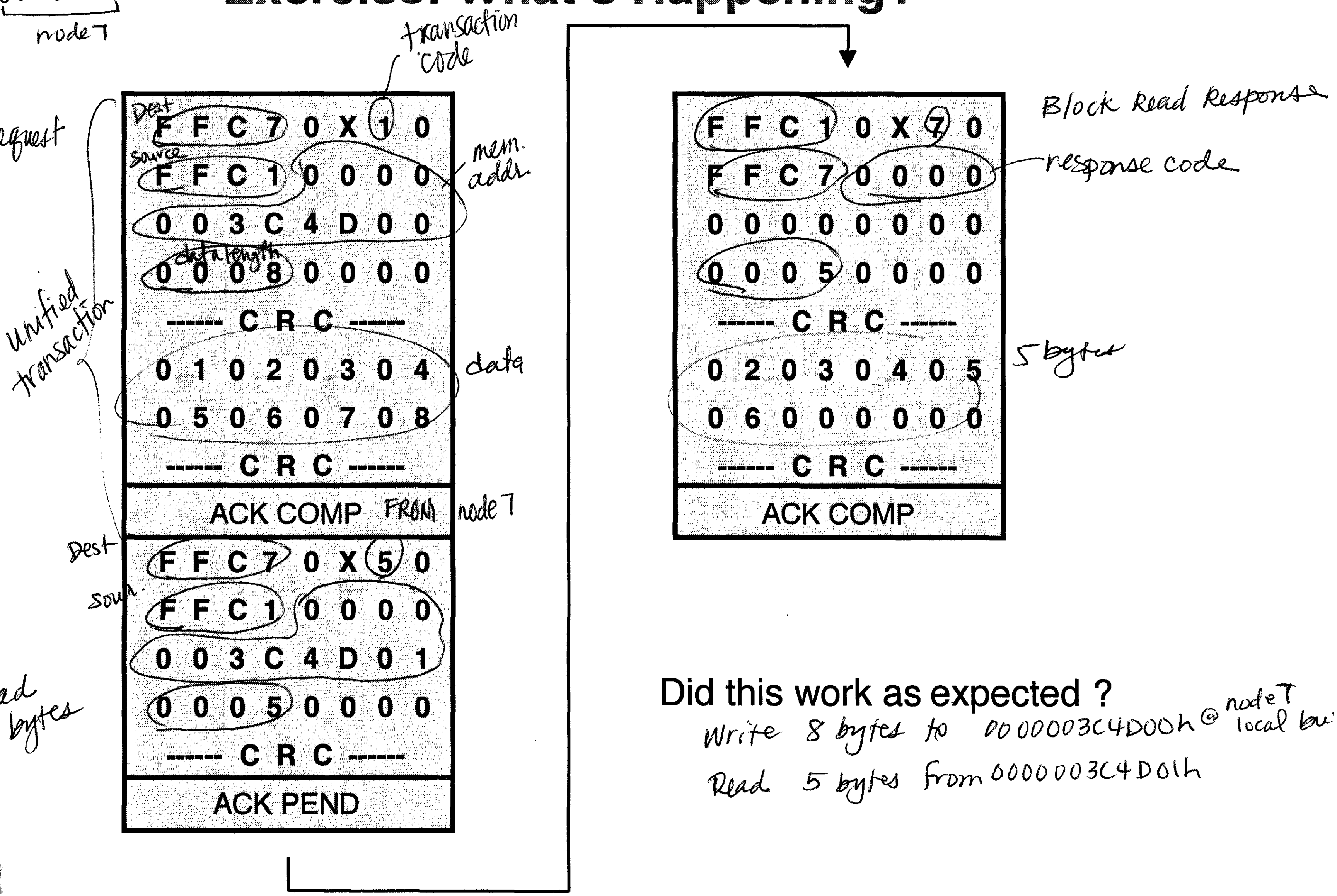
Block Wr. Request

Dest

00	01
1	02
2	03
3	04
4	05
5	06
6	07
7	08
ddr	data

BlockRead Req

read 5 bytes



```
F F C 7 0 X 1 0
F F C 1 0 0 0 0
0 0 3 C 4 D 0 0
0 0 0 8 0 0 0 0
------ C R C ------
0 1 0 2 0 3 0 4
0 5 0 6 0 7 0 8
------ C R C ------
ACK COMP
F F C 7 0 X 5 0
F F C 1 0 0 0 0
0 0 3 C 4 D 0 1
0 0 0 5 0 0 0 0
------ C R C ------
ACK PEND
```

```
F F C 1 0 X 7 0
F F C 7 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 5 0 0 0 0
------ C R C ------
0 2 0 3 0 4 0 5
0 6 0 0 0 0 0 0
------ C R C ------
ACK COMP
```

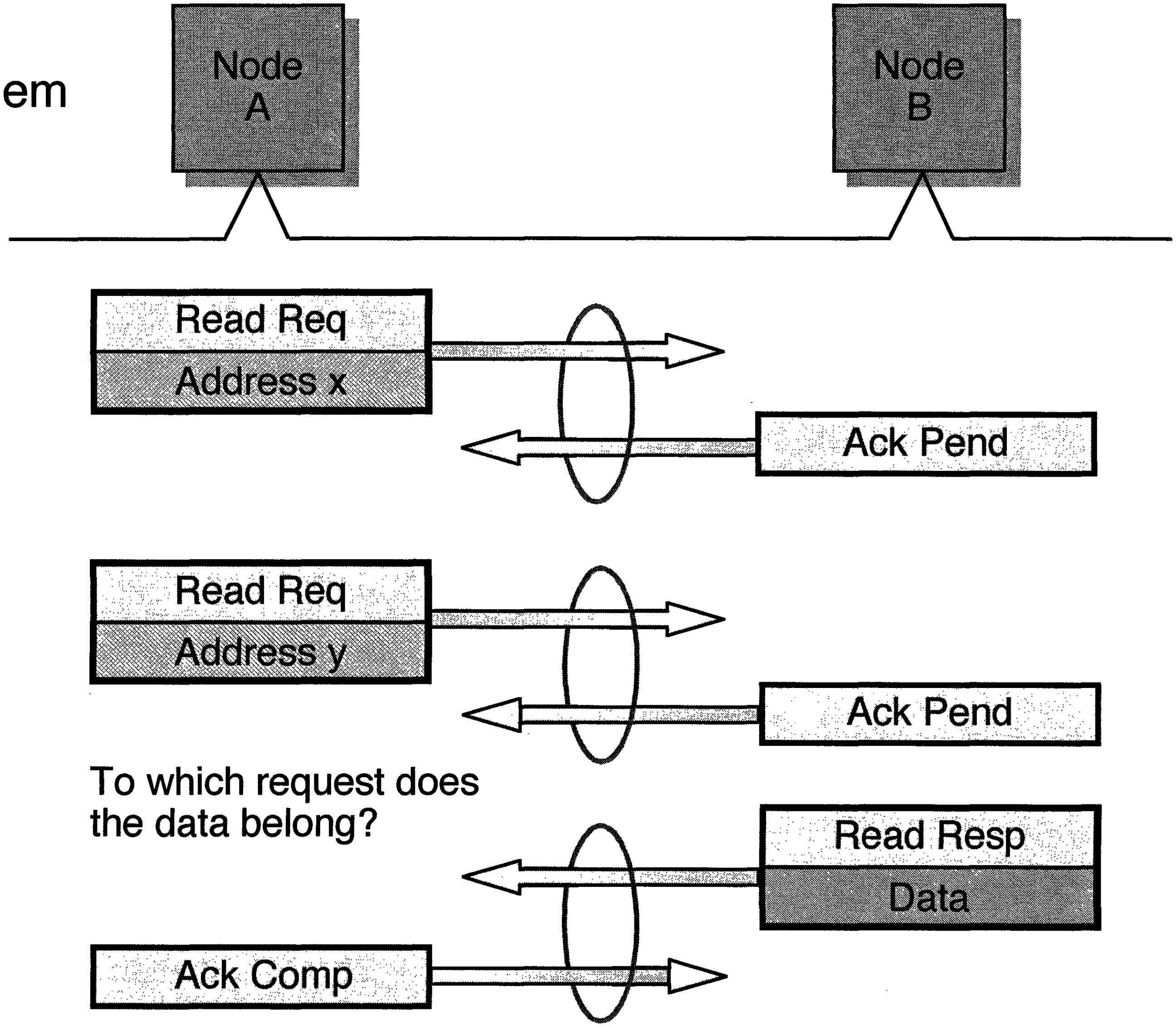
Did this work as expected ?

Write 8 bytes to 0000003C4D00h @ node7 local bu

Read 5 bytes from 0000003C4D01h

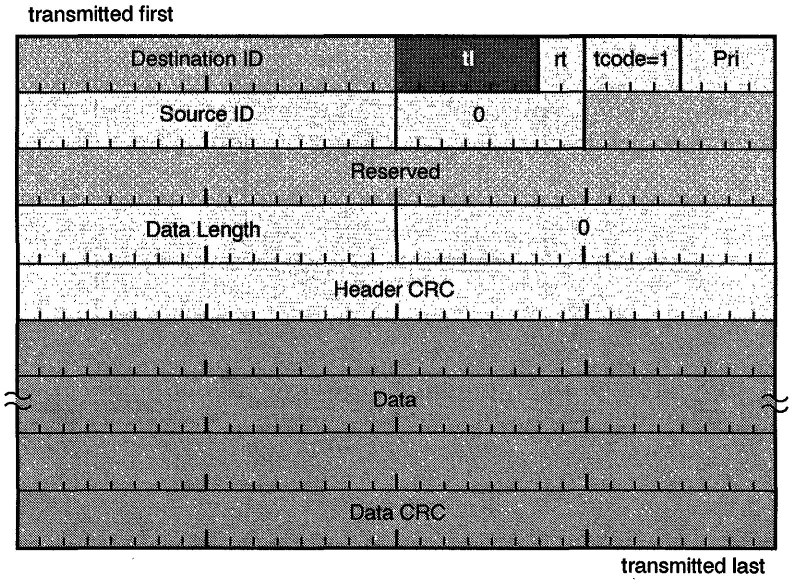
Which Data Is This Anyway?

The Problem



To which request does the data belong?

Transaction Labels (tl)



6 bit field

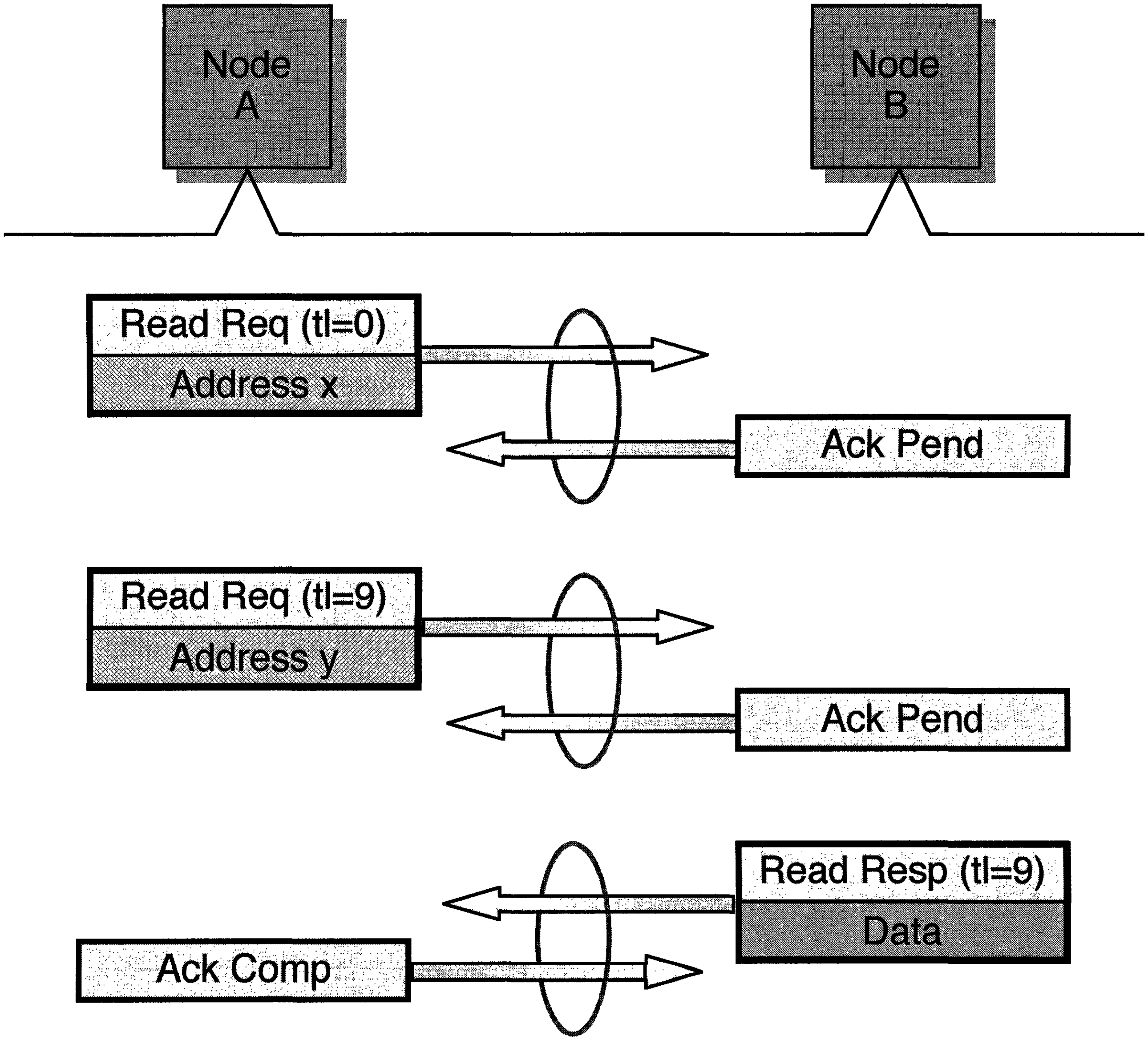
Unique for each outstanding operation between a pair of nodes

Sent in Request packet

Returned in corresponding Response packet

Requester uses it to match Response to Request

Transaction Labels



Single Data Quadlet Packets

Data being read/written is always 1 quadlet (32 bits)

No Data Length field

Single CRC for Header & Data

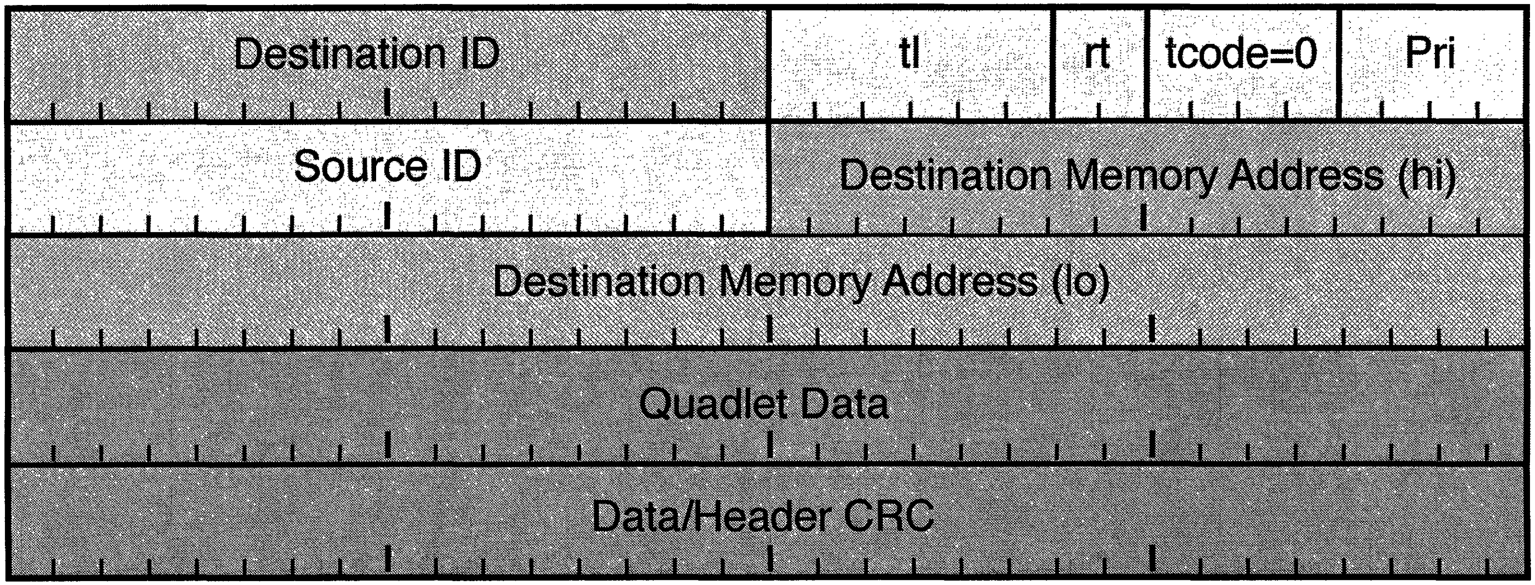
shorter packets

Required for certain register operations

Well suited to "Virtual Registers" implemented by microprocessor

Write Request for Single Data Quadlet

transmitted first



Destination ID | tl | rt | tcode=0 | Pri

Source ID | Destination Memory Address (hi)

Destination Memory Address (lo)

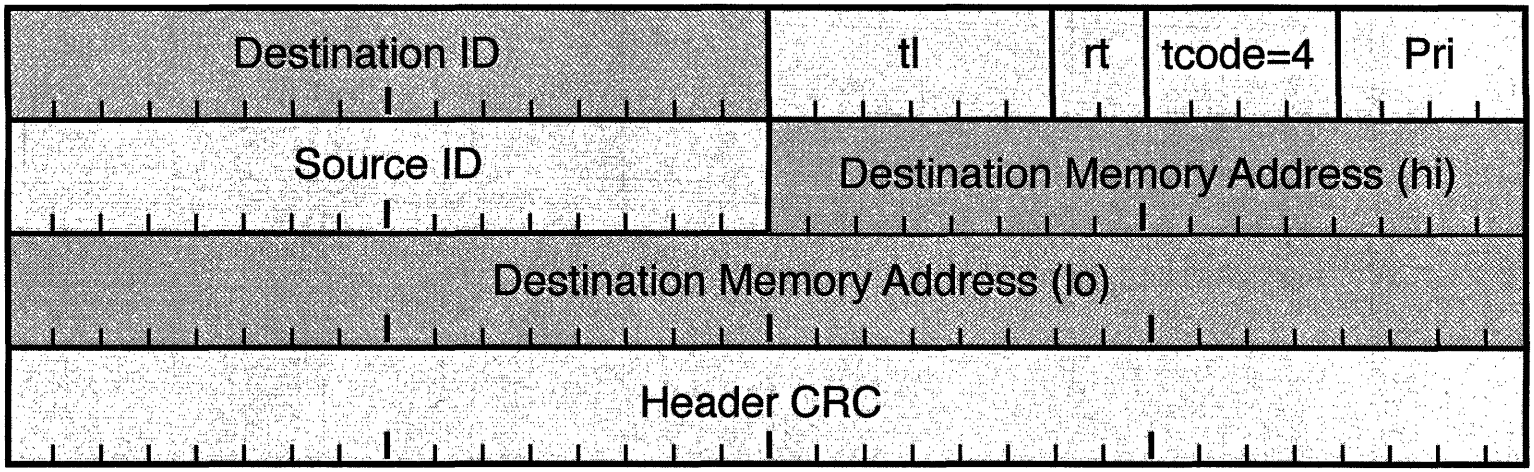
Quadlet Data

Data/Header CRC

transmitted last

Read Request for Single Data Quadlet

transmitted first



Destination ID | tl | rt | tcode=4 | Pri

Source ID | Destination Memory Address (hi)

Destination Memory Address (lo)

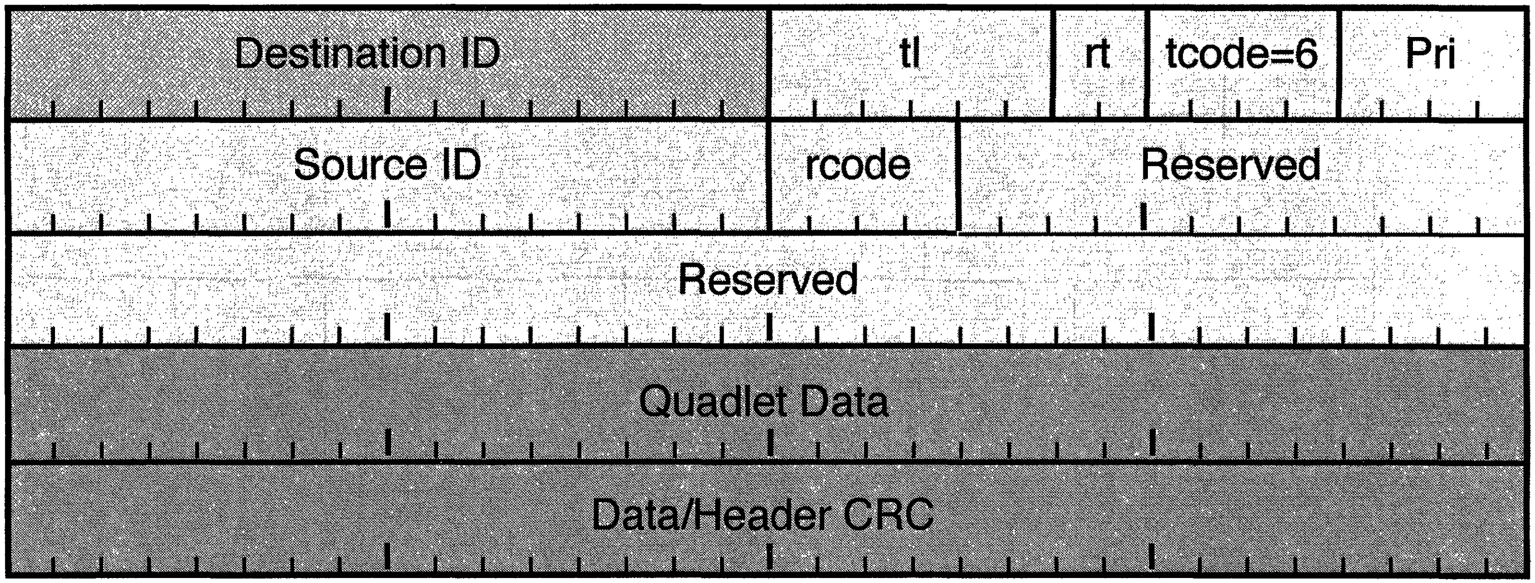
Header CRC

transmitted last

this page is intentionally blank

Read Response for Single Data Quadlet

transmitted first



Destination ID | tl | rt | tcode=6 | Pri

Source ID | rcode | Reserved

Reserved

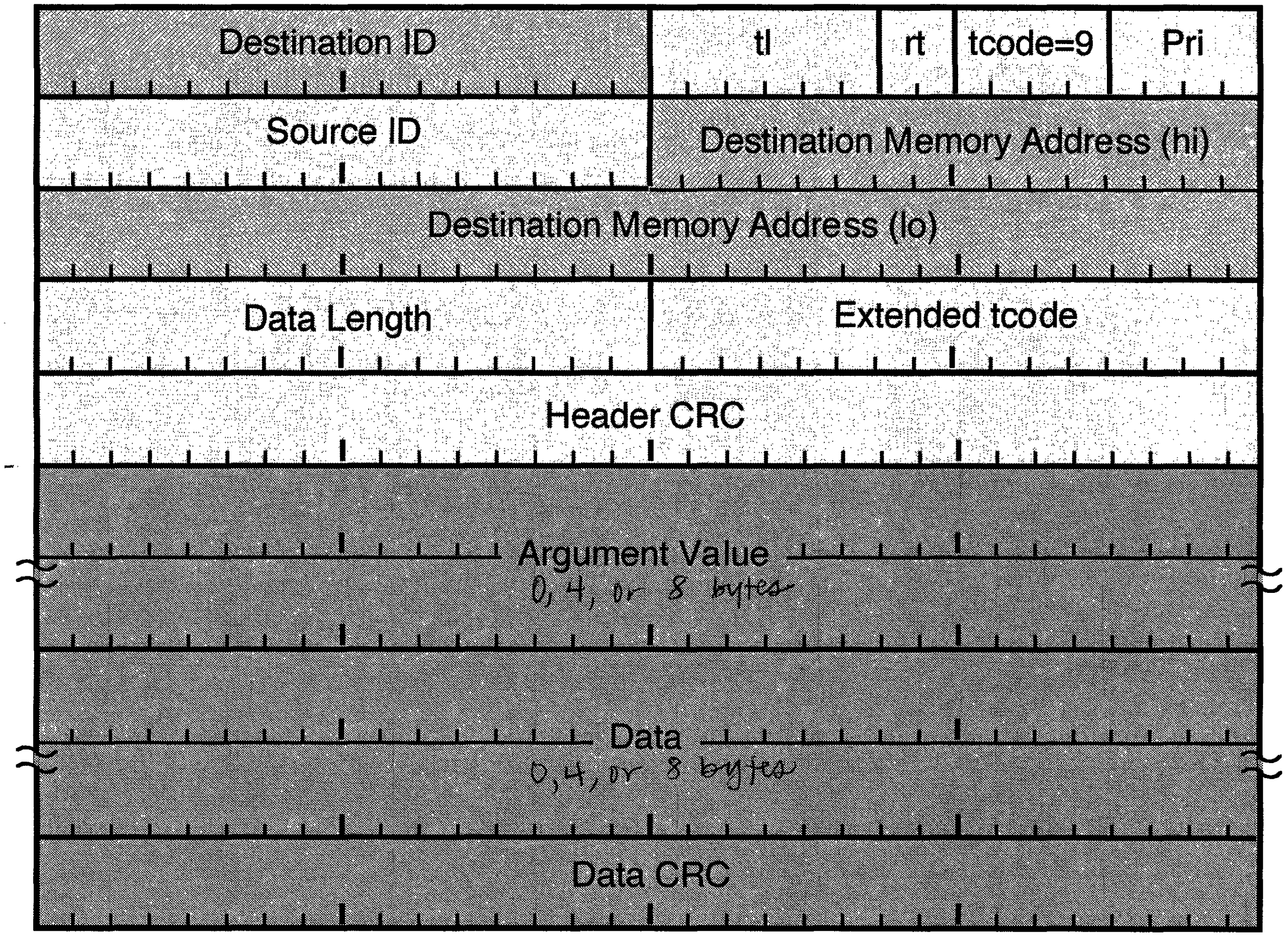
Quadlet Data

Data/Header CRC

transmitted last

Lock-Request Packet Format

transmitted first



Destination ID | tl | rt | tcode=9 | Pri

Source ID | Destination Memory Address (hi)

Destination Memory Address (lo)

Data Length | Extended tcode

Header CRC

Argument Value

0, 4, or 8 bytes

Data

0, 4, or 8 bytes

Data CRC

transmitted last

Lock Packet Definitions

Destination ID	16 bit address of receiving node
Source ID	16 bit address of sending node
tl	Transaction label
rt	Retry Code
tcode = 9	Transaction Code, 9 = Lock request
Pri	Priority, valid only in backplane environment
Data Length	Quantity of bytes in Argument Value and Data Fields
Extended tcode	Identifies the lock subcommand, 2 = compare and swap
Argument value	Data to compare with memory data
Data	Contents to write to memory if compare is successful

V

For More Information: Extended tcode Function

Extended tcode	Function	Definition
1h	MASK_SWAP	new_value = data_value I (old_value & ~arg_value);

2h	COMPARE_SWAP	if (old_value == arg_value)
		new_value = data_value;
		else new value = old_value;

3h	FETCH_ADD	new_value = old_value + data_value;
4h	LITTLE_ADD (little endian)	new_value = LittleEndAdd (old_value, data_value);

5h	BOUNDED_ADD
	(unequal add)
		if (old_value != arg_value)
		new_value = old_value + data value;
		else new value = old_value;

6h	WRAP_ADD	if (old_value != arg_value)
		new_value = old_value + data_value;
		else new_value = data_value;

7h	Vendor specific	

For More Information: Lock Transaction Data Length Parameter

Data Length = Argument Value Length + Data Value Length

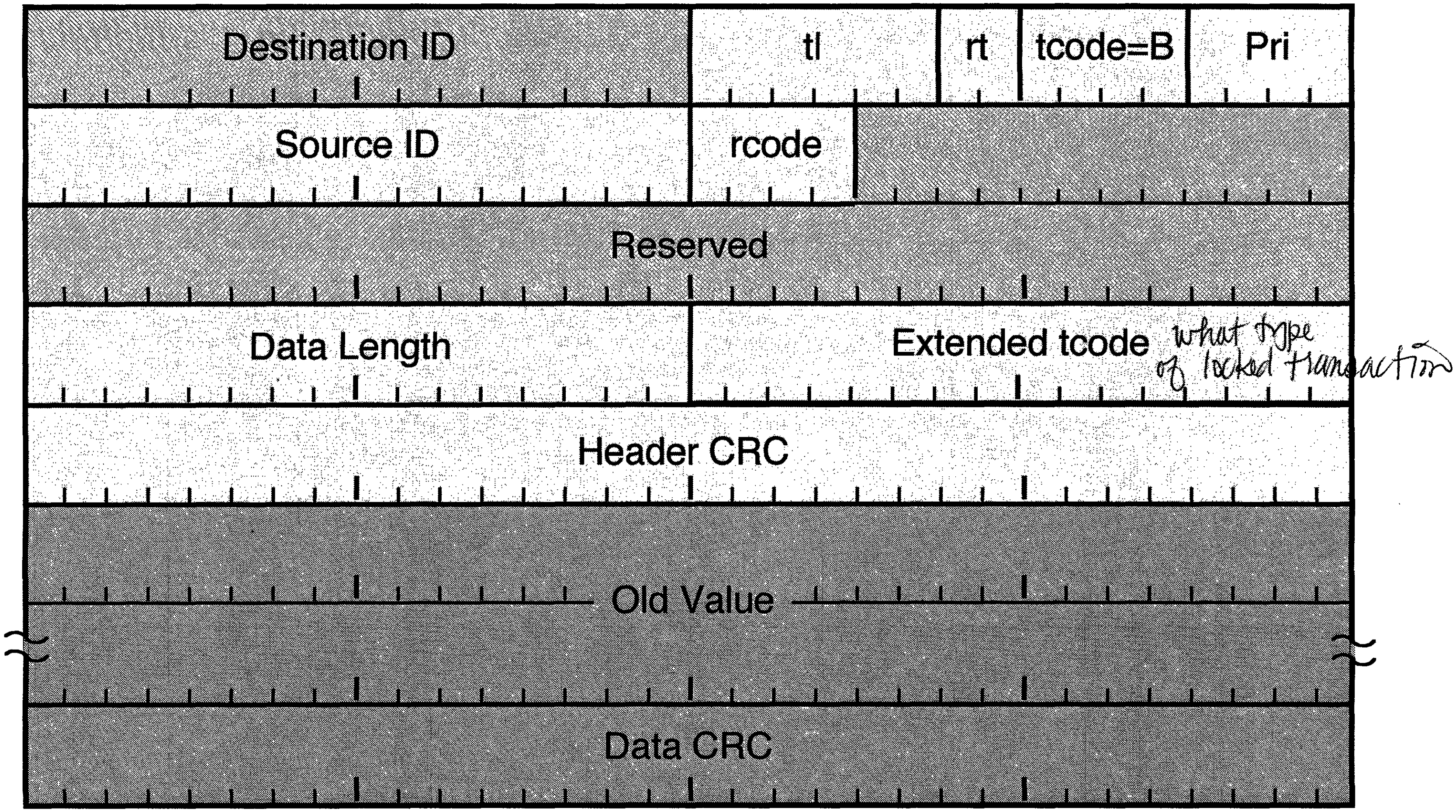
Only Data Lengths of 4, 8, and 16 Bytes supported

Argument Value Length and Data Value Length Depend on Function

Data Length (Bytes)	Function (extended tcode)	Data Value Length (Bytes)	Arg Value Length (Bytes)
4	FETCH_ADD LITTLE_ADD	4	0
8	MASK_SWAP, **COMPARE_SWAP** BOUNDED_ADD, WRAP_ADD	4	4
8	FETCH_ADD LITTLE-ADD (64 bit)	8	0
16	MASK_SWAP, **COMPARE_SWAP** BOUNDED_ADD, WRAP_ADD	8	8

Lock-Response Packet Format

transmitted first



Destination ID | tl | rt | tcode=B | Pri

Source ID | rcode

Reserved

Data Length | Extended tcode what type of locked transaction

Header CRC

Old Value

Data CRC

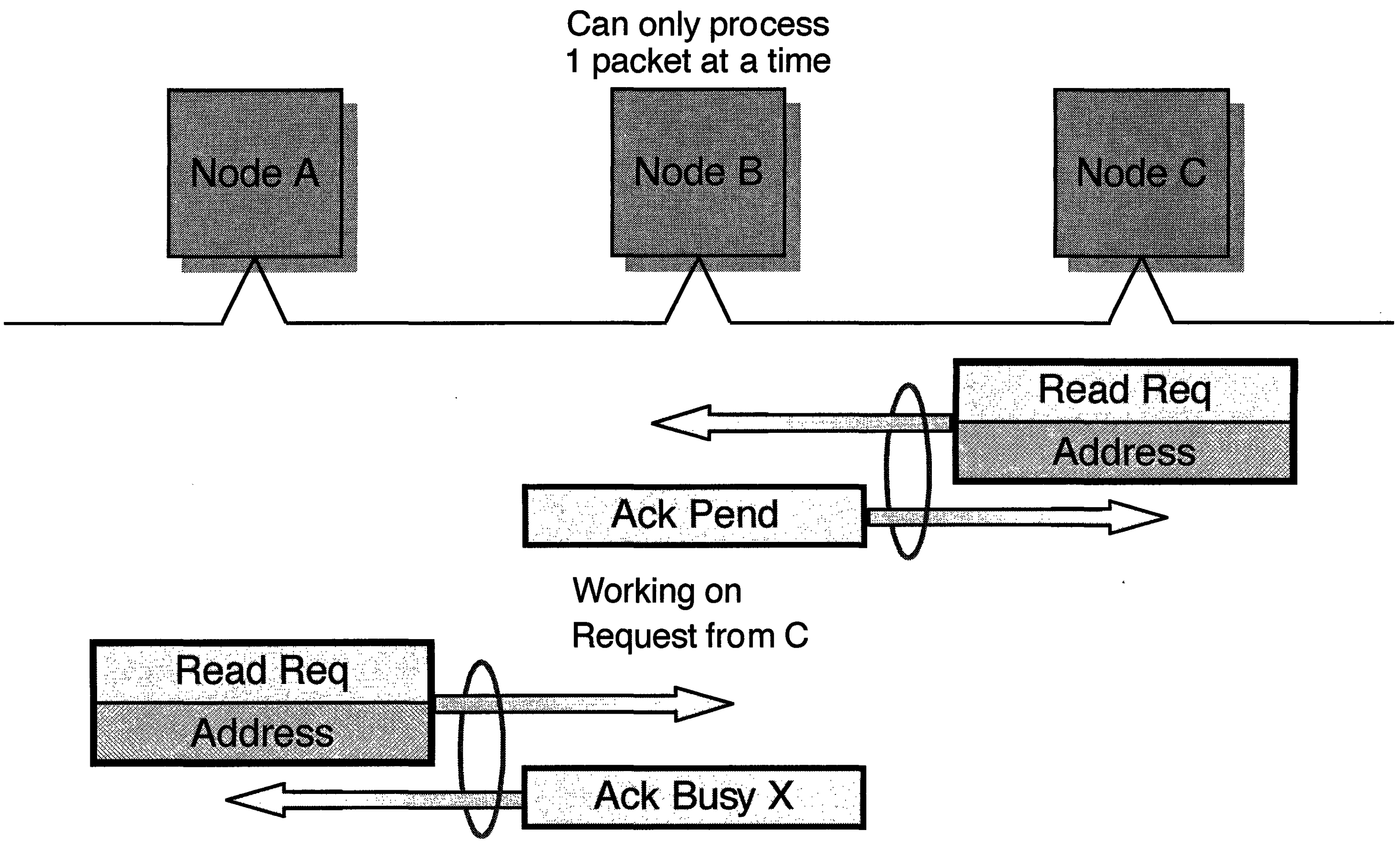
transmitted last

Lock Response Pack Format Definitions

Destination ID	High order 16 bits of address designating receiving node.
Source ID	High order 16 bits of sending node
tl	Transaction label
rt	Retry Code
tcode = B	Transaction code, B = LOCK RESPONSE
Pri	Priority, meaningful on backplane implementations only
rcode	Response Code
Data Length	Number of bytes in data field
Extended tcode	Specific lock function, 2 = compare and swap
Header CRC	32 bit Cyclic Redundancy Check for header quadlets.
Data CRC	32 bit Cyclic Redundancy Check for data quadlets.
Old Value	Data that was in referenced memory location of the selected node prior to lock operation

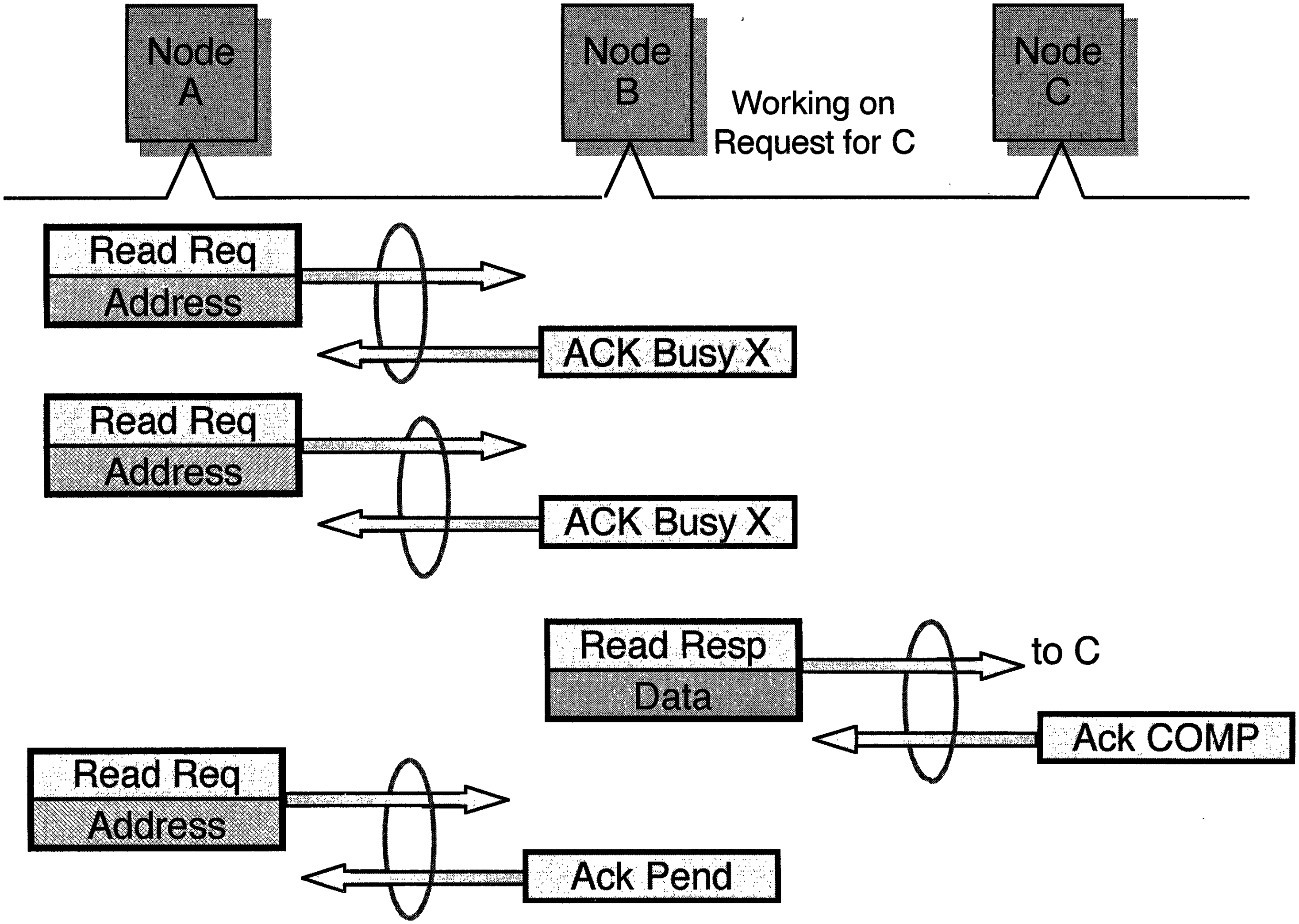
V

What If Packets Arrive Faster Than You Can Handle Them?



What do you do when bounced with a busy?

Simple Retry



Busy Options

Queue Packets

- Still need Busy for when Queue is full

Single Phase Retry

- When packet can't be processed - return ACK BUSY X
- Scheme on previous page
- Simple to implement
- High Priority devices hog the busy node

Dual Phase Retry

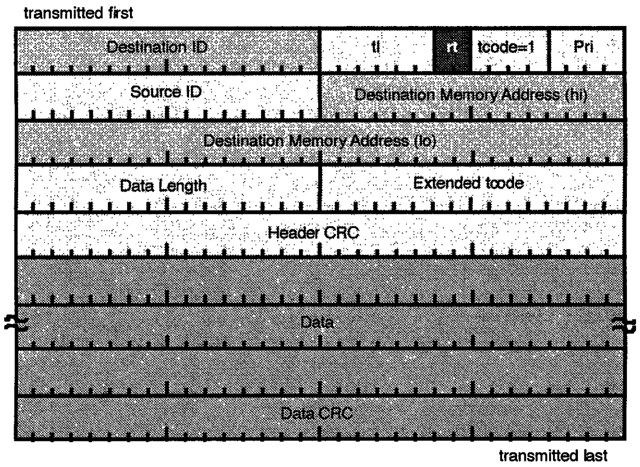
- Fairness mode - make sure all bounced devices get a chance

Busy Retry Management

Requester Retry Codes	Responder ACK Busy Codes
Retry_1 Retry_X Retry_A Retry_B	ACK Busy_X ACK Busy_A ACK Busy_B

Single phase equipment uses only Retry X and Busy X

Retry Code (rt)



rt code		name	meaning
0 0	0 h	Retry_1	Reservation Requested
0 1	1 h	Retry_X	No Reservation Requested
1 0	2 h	Retry_A	Used on next retry after ACK Busy_A Only used in dual phase retry
1 1	3 h	Retry_B	Used on next retry after ACK Busy_B Only used in dual phase retry

Dual Phase Retry Overview

Triggered by “bouncing” a packet twice

Packet receiver goes into a special mode

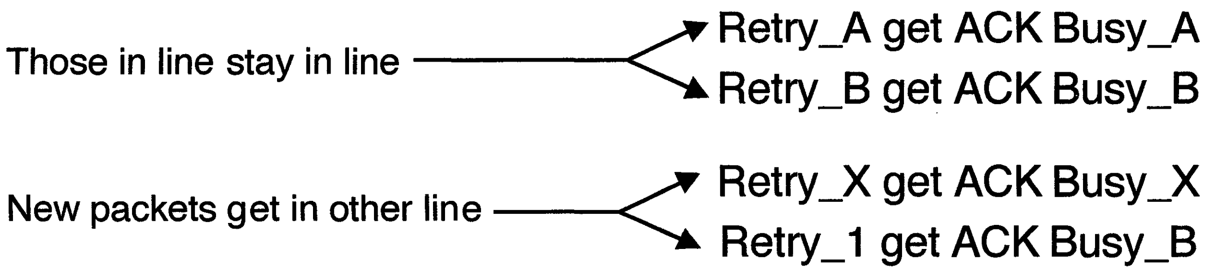
- Divides all senders into two groups (A and B)
- Only works on one group (A or B) at a time
- Keeps servicing a group until it’s empty
- All new packets are always put in the other group (B or A)

Nobody keeps count or keeps track of the groups

- Accomplished by ACK and Retry codes

Dual Phase Retry Management

Outbound Device (Requester)	Inbound Device (Responder)
All new packets coded Retry_X	If busy - respond ACK Busy_X
Re-send one packet with Retry_1 Go to service Retry_A only mode	If still busy - respond ACK Busy_A



Those in line stay in line

Retry_A get ACK Busy_A

Retry_B get ACK Busy_B

New packets get in other line

Retry_X get ACK Busy_X

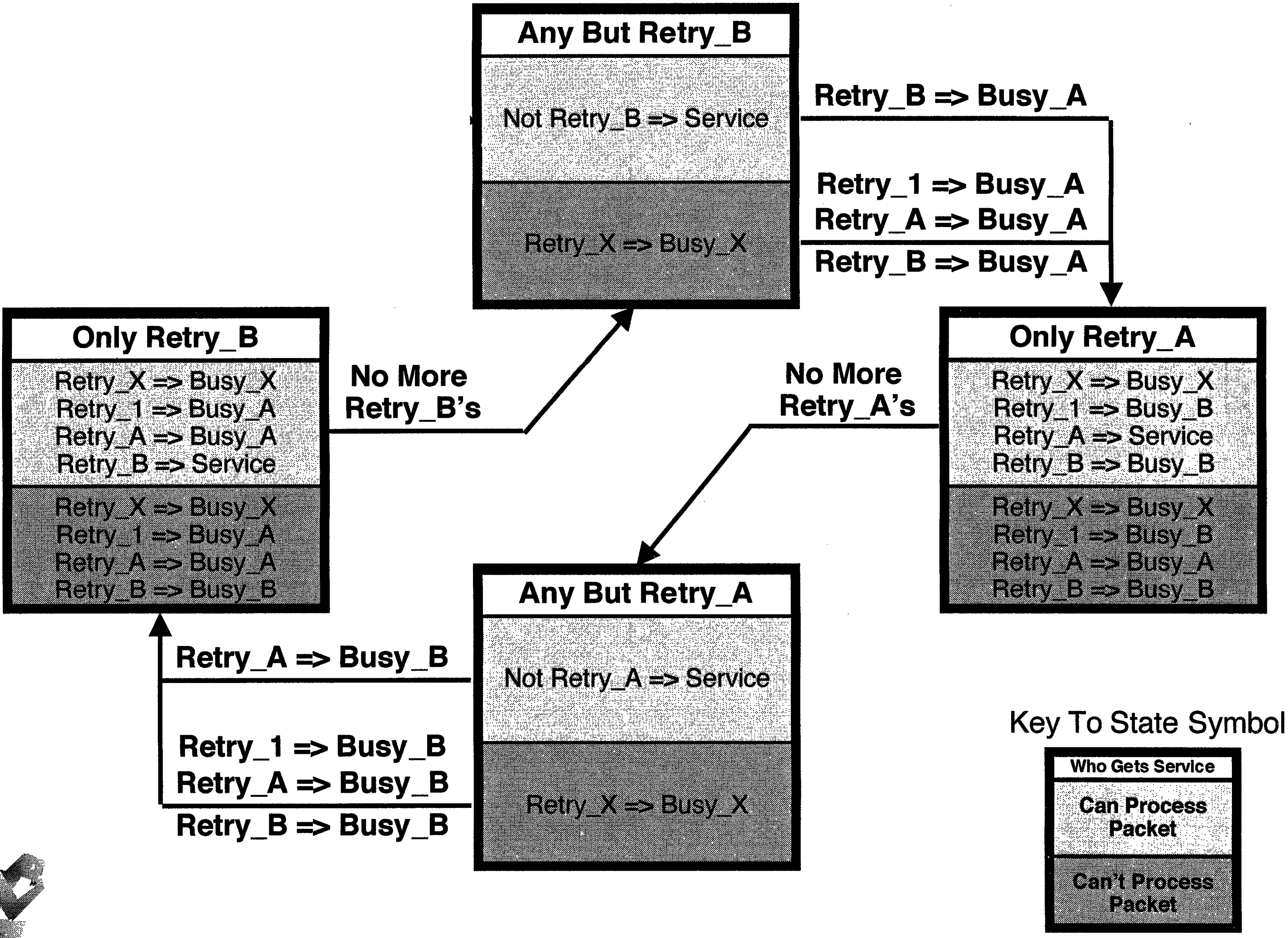
Retry_1 get ACK Busy_B

Service only Retry_A packets

Stay in mode until no more Retry_A

Repeat above but for B group

Dual Phase Retry Inbound Strategy State Diagram



Any But Retry_B

Not Retry_B => Service

Retry_X => Busy_X

Retry_B => Busy_A

Retry_1 => Busy_A
Retry_A => Busy_A

Retry_B => Busy_A

Only Retry_B

Retry_X => Busy_X
Retry_1 => Busy_A
Retry_A => Busy_A
Retry_B => Service

Retry_X => Busy_X
Retry_1 => Busy_A
Retry_A => Busy_A
Retry_B => Busy_B

No More Retry_B's

No More Retry_A's

Only Retry_A

Retry_X => Busy_X
Retry_1 => Busy_B
Retry_A => Service
Retry_B => Busy_B

Retry_X => Busy_X
Retry_1 => Busy_B
Retry_A => Busy_A
Retry_B => Busy_B

Any But Retry_A

Not Retry_A => Service

Retry_X => Busy_X

Retry_A => Busy_B

Retry_1 => Busy_B
Retry_A => Busy_B

Retry_B => Busy_B

Key To State Symbol

Who Gets Service

Can Process Packet

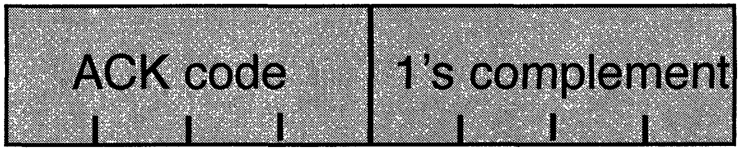
Can't Process Packet

Dual Phase Retry Codes

Subaction Age	Prior ack code	Retry Code	
		Single Phase	Dual Phase
Not Oldest	—	Retry X	Retry X
	ack Busy X		
	ack Busy A		
	ack Busy B		
~~Oldest~~ (highest priority)	—	Retry X	Retry 1
	ack Busy X		
	ack Busy A		Retry A
	ack Busy B		Retry B

Only one request and one response per talker can be oldest. Selection of oldest is implementation dependent.

Acknowledge Formats



ACK code | 1's complement

ACK Code		Name
0000	0h	Reserved
0001	1h	ACK Complete
0010	2h	ACK Pending
0011	3h	Reserved
0100	4h	ACK Busy_X
0101	5h	ACK Busy_A
0110	6h	ACK Busy_B
0111	7h	Reserved
1000	8h	Reserved
1001	9h	Reserved
1010	Ah	Reserved
1011	Bh	ACK Tardy Low power
1100	Ch	ACK Conflict error
1101	Dh	ACK Data error
1110	Eh	ACK Type error
1111	Fh	ACK Address Error

Review

1. How does the “talking” node on “listening” node differentiate between Write Request, Write Response, Read Request, Read Response, Lock Request and Lock Response?
2. What does each of the above transactions do?
3. Why do the Acks not have an address?
4. How many address bits are required in 1394 addressing to identify a register location in one node of seven on a single bus? On a 1394 network with 62 busses?

Asynchronous Notes

Asynchronous Notes

Section 3

Control and Status Registers (CSRs)

Subjects Covered

1394 Register Space

Core Registers

Bus Dependent Registers

Configuration ROM

What is a Register?

A register is a place for storing information

- A Guest Register in a hotel or wedding is a book where guests write their name
- A Computer Register is normally a latch or group of latches
- A CSR is a fixed memory location in each node that keeps information describing that node

Computer Registers and CSRs are generally volatile - they lose their contents on reset or power off

ROM is non volatile

What are CSRs?

Control and Status Registers

A defined set of registers in a memory mapped address space intended to be used as part of an open interface

Defines both a register set and a configuration ROM

Used by 1394, SCI (Scaleable Coherent Interface), NuBus (Texas Instruments), Multibus II (Intel) same registers & locations

Defined in ISO/IEC 13213 and ANSI/IEEE 1212
document 1394 built on.

CSR Registers

Registers are 4 bytes (32 bits) or 8 bytes (64 bits) wide

Registers are addressed by their offset from the initial register space or other base address

Most registers are optional and there is a large area for vendor specified registers or bus dependent information

Initial contents of each register is defined by spec

Results of a read or a write is defined by spec

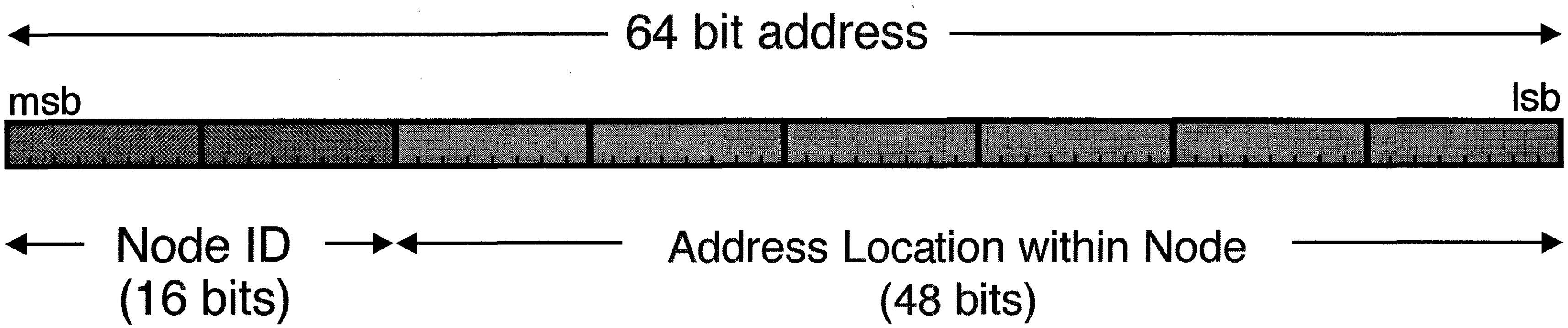
Register locations are “Well Known Addresses” so other nodes can read or write to them.

1394 Addressing

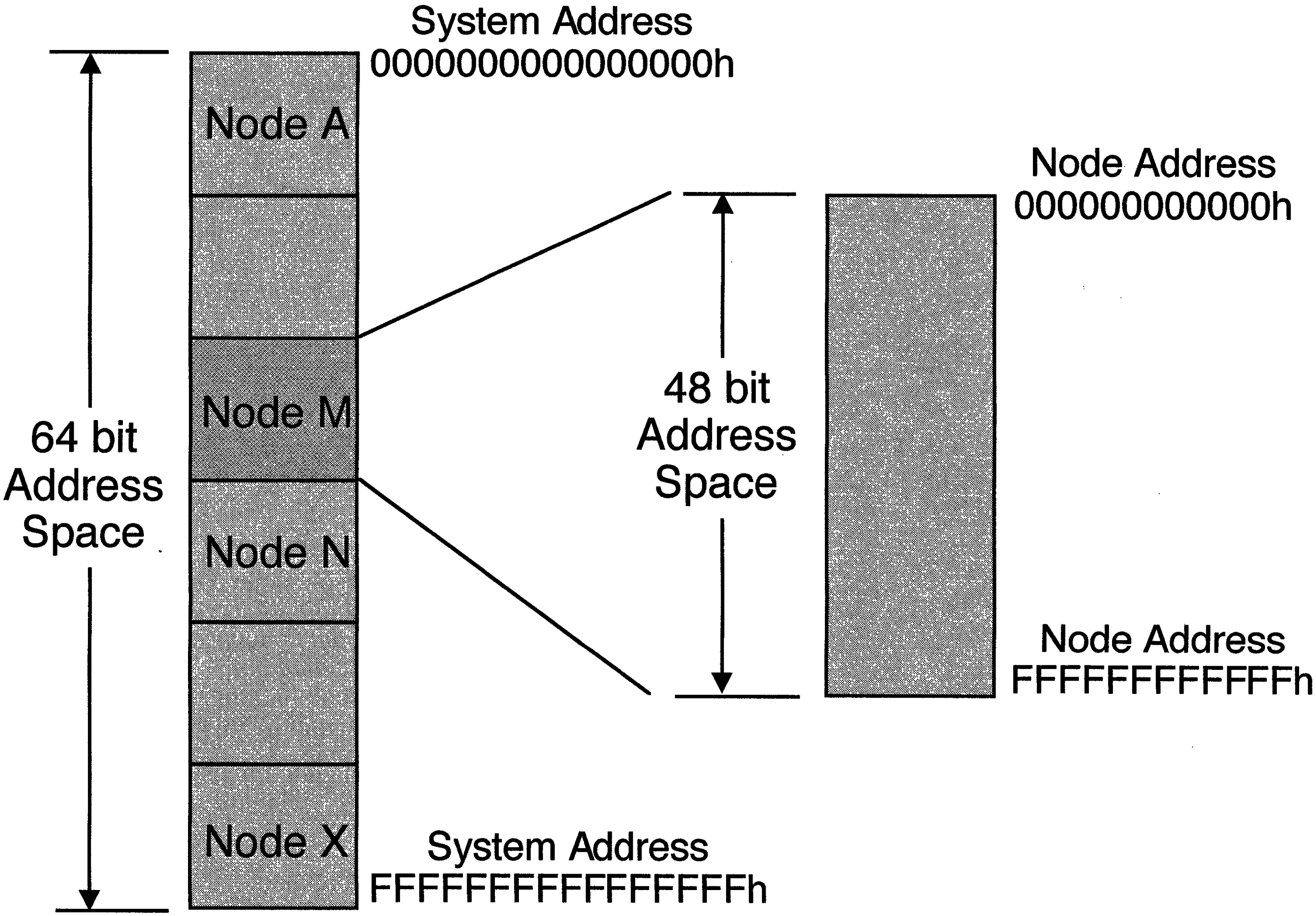
Packets use 64 bit Addresses

Top 16 bits determine Node

Bottom 48 bits address location within Node

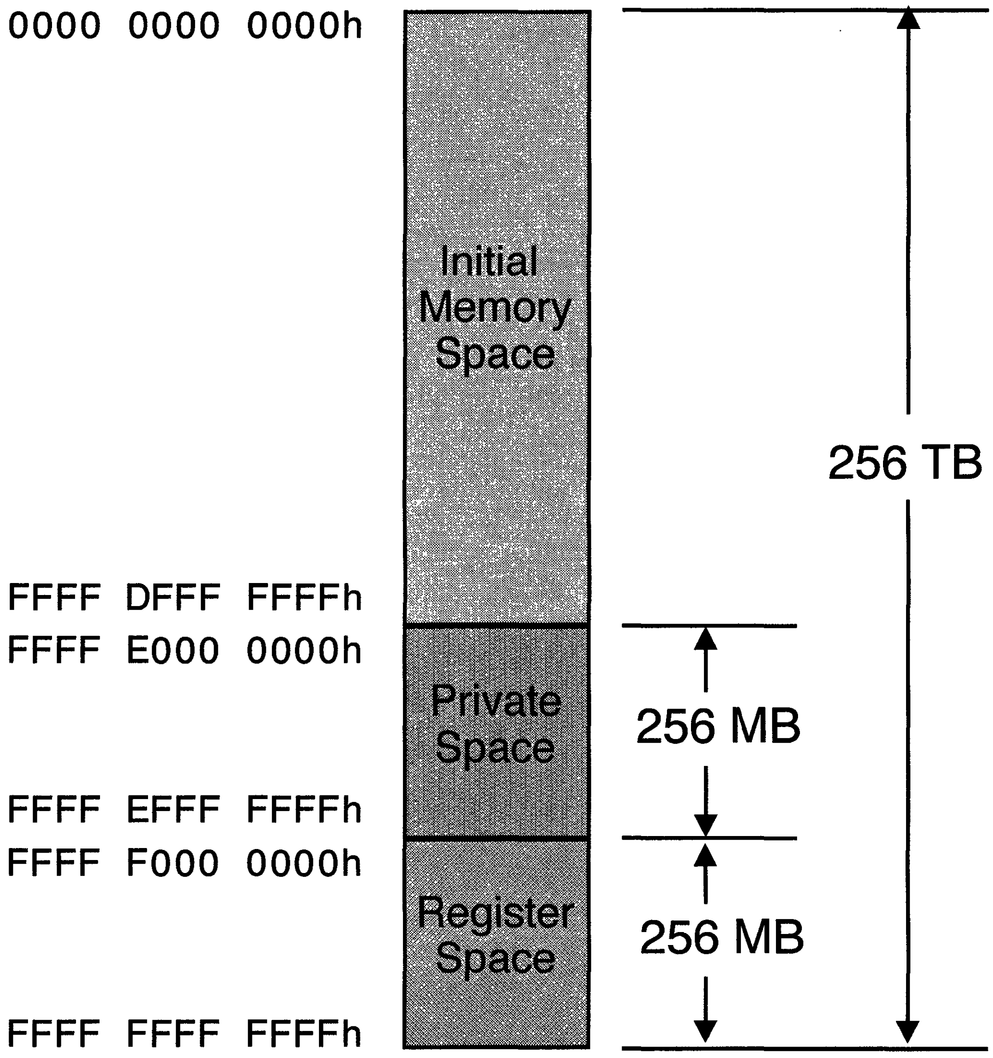


1394 Address Map



Addresses are shown from top - down throughout this course

Node Space Addressing



0000 0000 0000h

Initial Memory Space

256 TB

FFFF DFFF FFFFh

FFFF E000 0000h

Private Space

256 MB

FFFF EFFF FFFFh

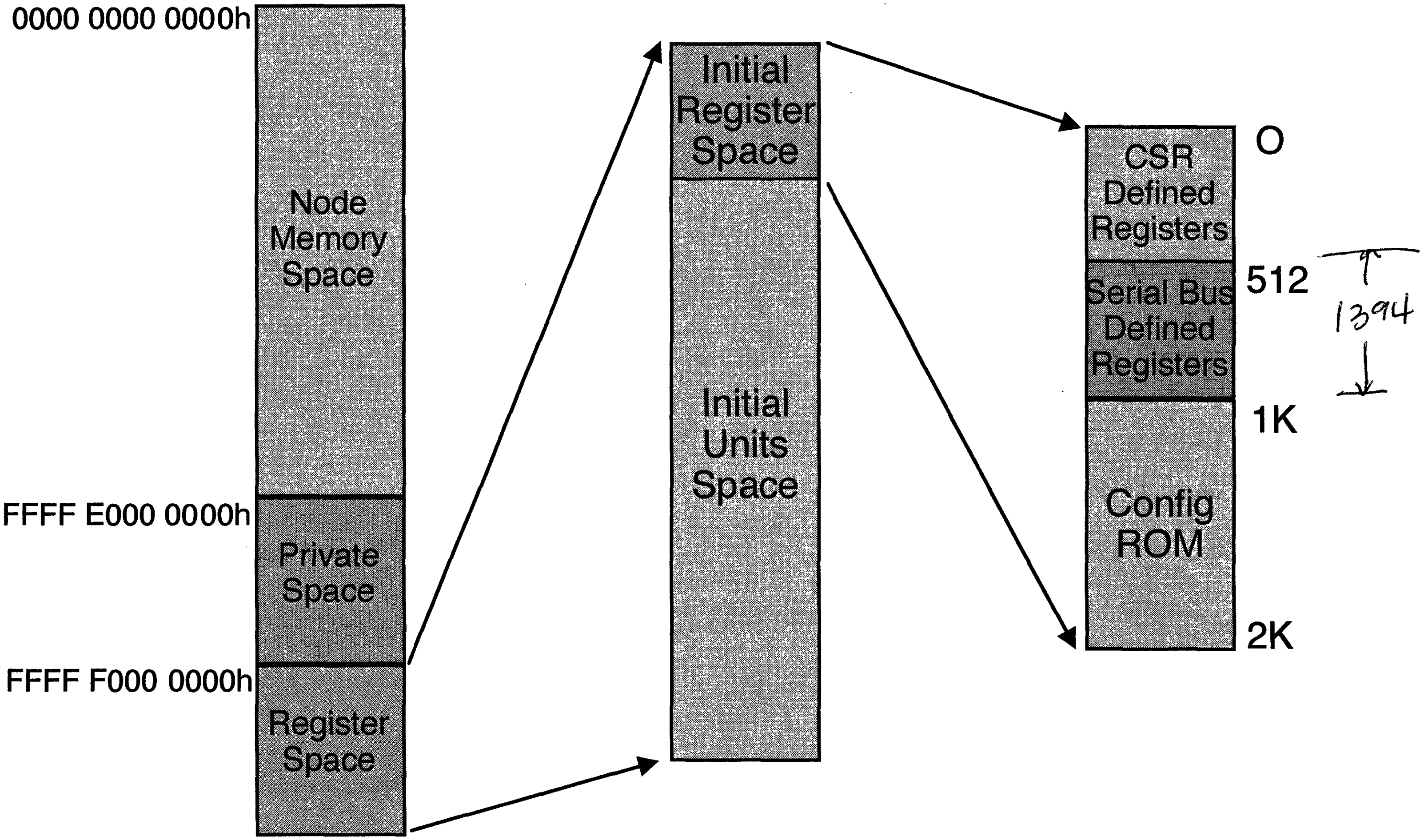
FFFF F000 0000h

Register Space

256 MB

FFFF FFFF FFFFh

Register Space



Registers Base Address = FFFF F000 0000h.

Register Structure

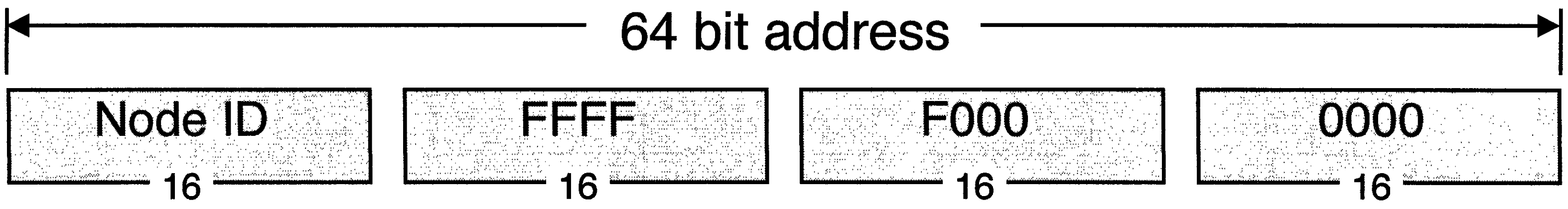
Each register has defined bits, reserved bits, and vendor specific bits

Some registers are read only (RO),
some are read/write (RW) and
some are write only (WO)

Some registers are limited to Quadlet read only or lock Transaction for access

In register space, there can be side effects from write Transactions

Addressing CSRs



64 bit address

Node ID	FFFF	F000	0000
16	16	16	16

Example: To address Bus Manager CSR (addr 021Ch) on node FFC8h:

Register Space Offset	. . . .	FFFF	F000	0000
Bus Mngr CSR Offset	. . . .	. . . .	. . . .	021C
Node Address	FFC8	. . . .	. . . .	. . . .
Register address	FFC8	FFFF	F000	021C

Core Registers (defined by 1212)

0000	*State Clear
0004	*State Set
0008	*16 bit ID of this node
000C	*Reset Start
0018	*Split timeout, Integers of second
001C	*Split timeout, fractions of second
0020	Node Self Test Argument, Hi
0024	Node Self Test Argument, Lo
0028	Node Self Test, Start
002C	Node Self Test, Status
0050	Interrupt Target
0054	Interrupt Mask
0058 to 007C	Assorted clock control, normally not implemented on Serial Bus
0080 to 00FC	Message Request/Response

* required in 1394-1995 or SBP-2

this page is intentionally blank

State Register

Unit depend	Bus depend	Lost	Dreq	r	Elog	Atn	Off	State
16	8	1	1	1	1	1	1	2

A write of a 1 bit to STATE CLEAR register clears the identified bit.

A write of a 1 bit to STATE SET register sets the identified bit.

A read of either address 0 or 4 gives the contents of the State Register.

Λ

State Registers

Unit depend	Unit Vendor Specific
Bus depend	Bus type specific (see next slide for 1394)
Lost	Set on reset, cleared by the software; indicates the unit is “lost”
Dreq	Disable request from unreliable nodes
Elog	An error has been detected and the error log has been updated
Atn	Attention; this node should be prepared for on-line replacement
Off	Prevent node access while a board is being replaced
State	0 - Running 1 - Initializing 2 - Testing 3 - Dead

V

State Register Bus Dependent Information

Gone	r	r	r	r	Abdicate	Linkoff	CMSTR

Gone Set to a 1 on any reset, cleared when reset is completed

Linkoff Setting this bit powers off the Link layer (Defined in Implementation section)

CMSTR Node is Cycle Master Capable (defined in Isochronous section)

Abdicate After a bus reset, the incumbent Bus Manager will wait 125 mSec before doing a lock request to the Bus Manager ID CSR (1394a)

Serial Bus Defined Registers

Address(h)	Name	Description
0200	CYCLE_TIME	For isochornous services, counts 24.576 MHz clocks
0204	BUS_TIME	For synchronized bus time
0208	POWER_FAIL_IMMINENT	Power fail warning
020C	POWER SOURCE	Power fail warning
0210	BUSY_TIMEOUT number of retries to a busy node	For transaction capable nodes - limits
0218	PRIORITY BUDGET	For priority arbitration
021C	BUS_MANAGER_ID	For selecting or locating bus manager
0220	BANDWIDTH_AVAILABLE	Bandwidth allocation
0224-0228	CHANNELS_AVAILABLE	Channel allocation
022C	MAINT_CONTROL	Diagnostics, to generate specific errors
0230	MAINT_UTILITY	Diagnostics
0234	BROADCAST CHANNEL	For broadcast via asynchronous streams

Serial Bus Defined Registers in Initial Units Space

Offset (h)	Name	Notes
800 - 8FC		Reserved
900 - 9FC	PLUG CONTROL REGISTERS	Logical connections of isochronous devices IEC 61883
A00 - AFC		Reserved
B00 - CFC	FCP CMD Frame	IEC 61883
D00 - EFC	FCP RESP Frame	IEC 61883
F00 - FFC		Reserved
1000 - 13FC	TOPOLOGY MAP	Bus Manager only
1400 - 1FFC		Reserved
2000 - 2FFC	SPEED MAP	Bus Manager only (obsoleted)
3000 - FFFC		Reserved

TOPOLOGY MAP will be defined in the Bus Management section.

ROM Hierarchy

(Node address) FFFF F000 0400h (1K off Register Base address)

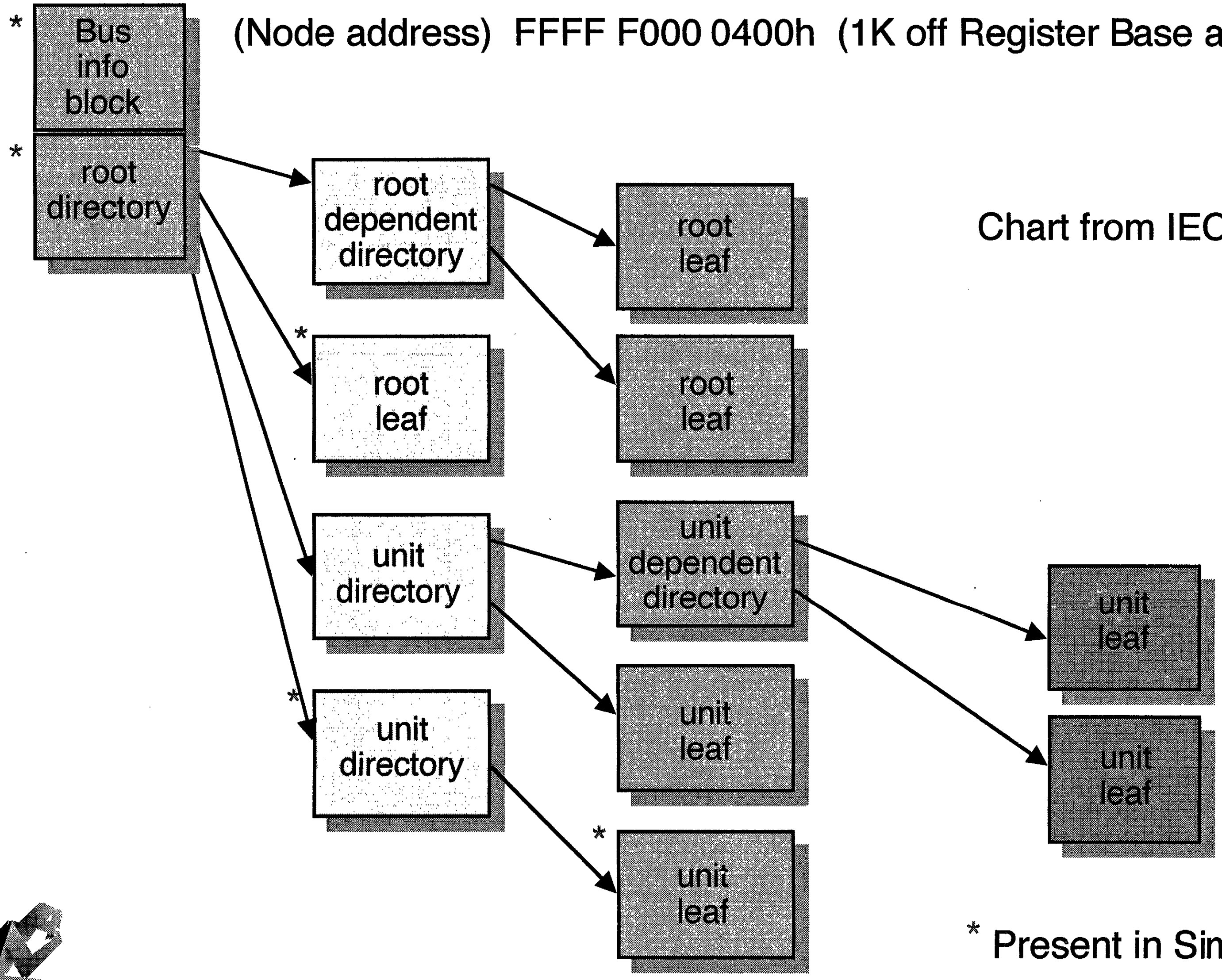
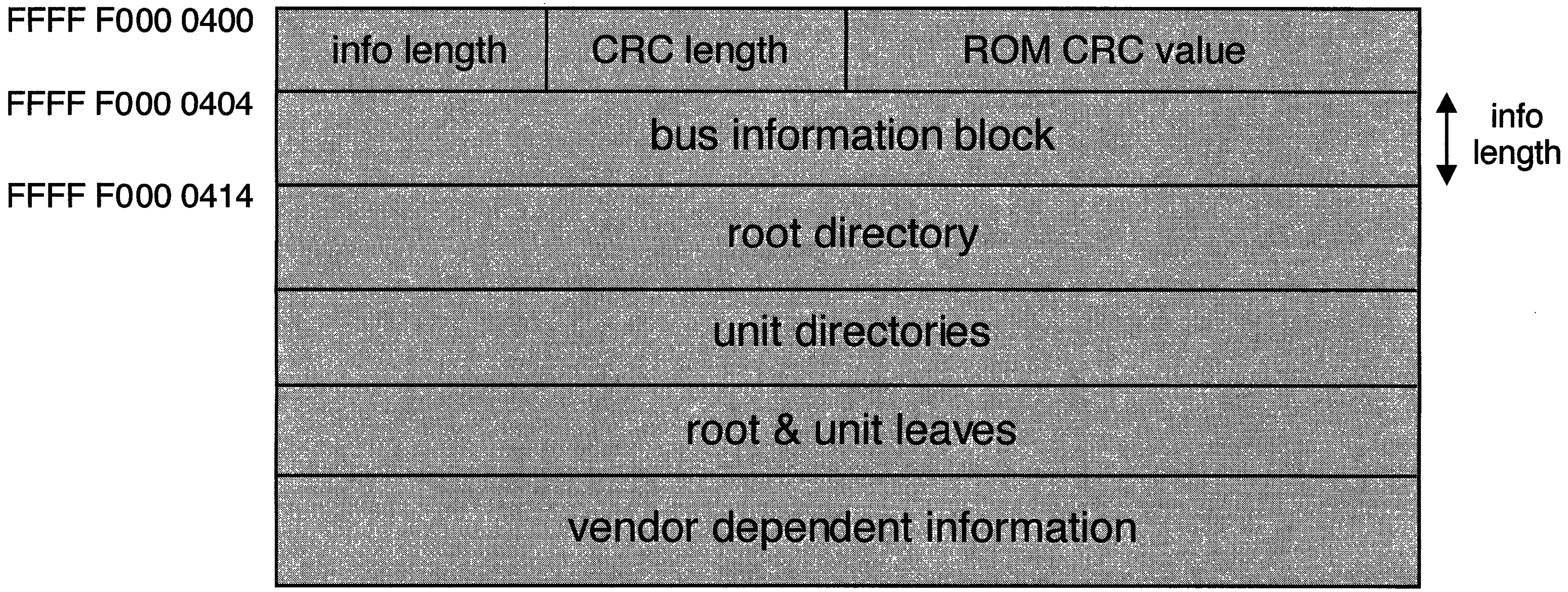


Chart from IEC 13213

* Present in Simple Devices

CSR General ROM format



FFFF F000 0400

info length	CRC length	ROM CRC value

FFFF F000 0404

bus information block

FFFF F000 0414

root directory

unit directories

root & unit leaves

vendor dependent information

Info length = number of quadlets in bus_info_block (always 4)

CRC length = quadlets of this ROM protected; minimum = bus info block,
maximum = 255

ROM CRC value = the 16 bit CRC check character for this ROM

this page is intentionally blank

Bus Info Block

<table>
<tr><td colspan="6">Length = 4</td><td>CRC Length</td><td colspan="6">CRC</td></tr>
<tr><td colspan="6">31h “1”</td><td>33h “3”</td><td colspan="3">39h “9”</td><td colspan="3">34h “4”</td></tr>
<tr><td>IR</td><td>C</td><td>I</td><td>B</td><td>P
M
C</td><td>Resv
(3 bits)</td><td>cyc clk acc</td><td>max rec
(4 bits)</td><td>r
(2 bits)</td><td>Max
ROM
(2 bits)</td><td>Generation #'s inc when changed
G
(4 bits)</td><td>r</td><td>Link
Speed
(3 bits)</td></tr>
<tr><td colspan="10">Node vendor ID</td><td colspan="3">config chng
chip ID high</td></tr>
<tr><td colspan="13">chip ID low</td></tr>
</table>

Bus Info Block

Bus Info Block

IR	Isochronous Resource Manager capable
C	Cycle master capable
I	Isochronous capable
B	Bus Manager capable
PMC	Power Manager Capable
G	Generation number - Indicates information in configuration or any leaf or directory changed
Link Speed	Maximum speed of the node's link layer
cyc clk acc	Accuracy of cycle clock in parts per million (1-100)
Node vendor ID	24 bit globally unique Organizationally Unique Identifier (OUI) assigned by IEEE Registration Authority
Chip ID Hi/Lo	40 bit globally unique ID administered by node vendor. Node vendor ID concatenated with chip ID Hi and Lo yield a 64 bit Extended Unique ID (EUI-64).
Max ROM	Defines alignment for block read requests to configuration ROM how much can it read @ a time

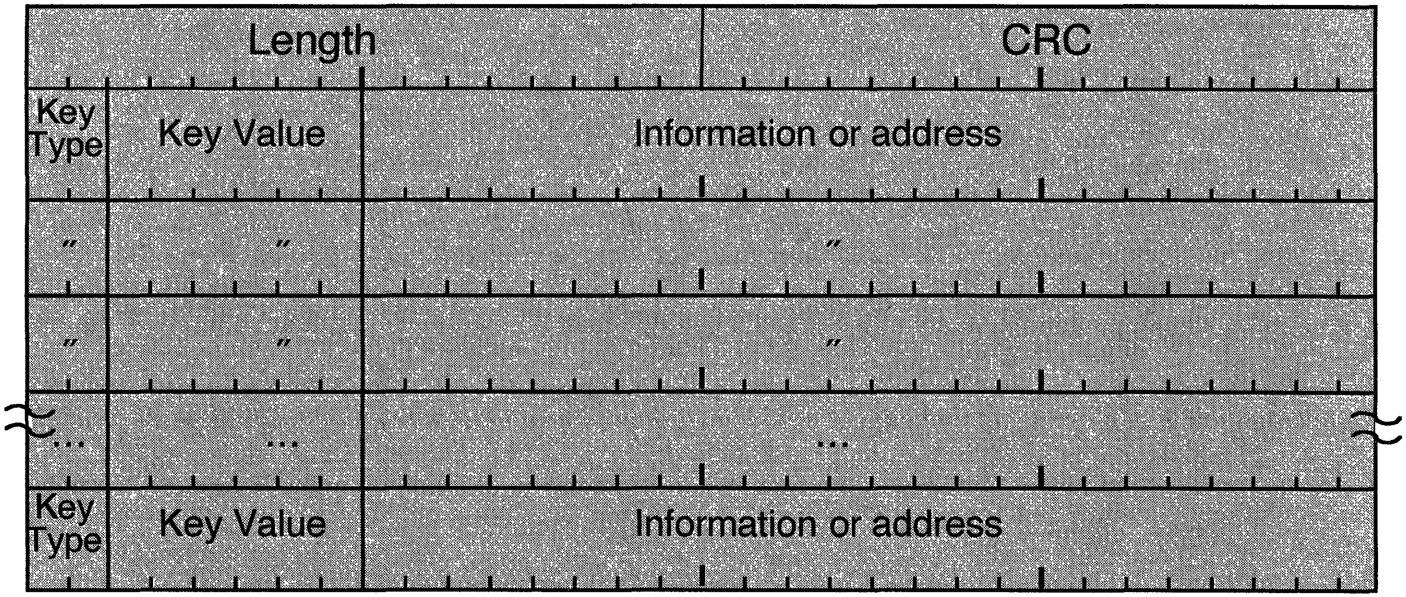
V

Maximum Record Length

max rec - the maximum of an asynchronous write addressed to this node

max rec	Max size in bytes
0h	not specified
1h	4
2h	8
3h	16
4h	32
5h	64
6h	128
7h	256
8h	512
9h	1024
Ah	2048
Bh	4096
Ch	8192
Dh	16384
Eh - Fh	reserved

Unit or Root Directory



Length		CRC
Key Type	Key Value	Information or address
"	"	"
"	"	"
...	...	...
Key Type	Key Value	Information or address

Key Type (2 bits)

- 00 - immediate value
- 01 - initial-register-space offset for an immediate value
- 10 - indirect-space offset for a leaf
- 11 - indirect-space offset for a directory

Key Value (6 bits)

identifies the 24 bit directory entry

Root Directory Example

LENGTH = 3	CRC
0C	00 83 D2
03	E4 71 04
8D	00 00 02

Note: These three entries are required by 1394-1995. Many others are permissible.

Configuration ROM Example

What type of entry is the first one?
Using the reference material in the back of this section, what capabilities are supported?

What type of entry is the second one?
What is the significance of the 24 bit entry?

What type of entry is the third one?
What steps would you use to find the leaf?

V

Questions???

What is the Node address of that node?

- Check the NODE_ID CSR

Does that node support split timeout?

- Check the NODE_CAPABILITIES.spt bit

Is that node doing a reset?

- Check the STATE_BITS.lost bit

What protocol does that node support?

- Check the bus_info_block of the CONFIGURATION ROM

I have a problem I need to notify somebody

- Set the STATE_BITS.elog bit

Review - CSR

You need to discover the node capabilities. What is the full chain of pointers to find that information?

You need to discover the unit's power requirements. What is the full chain of pointers to find that information?

Reference - Keyvalues

01	Textual_Descriptor
02	Bus_Dependent_Info
03	Module_Vendor_ID
04	Module_Hardware_Version
05	Module_Spec_ID
06	Module_Software_Version
07	Module_Dependent_Info
08	Node_Vendor_ID
09	Node_Hardware_Version
0A	Node_Spec_ID
0B	Node_Software_Version
0C	Node_Capabilities
0D	Node_Unique_ID
0E	Node_Units_Extent
0F	Node_Memory_Extent
10	Node_Dependent_Info
11	Unit_Directory
12	Unit_Spec_ID
13	Unit_Software_Version
14	Unit_Dependent_Info
15	Unit_Location
16	Unit_Poll_Mask

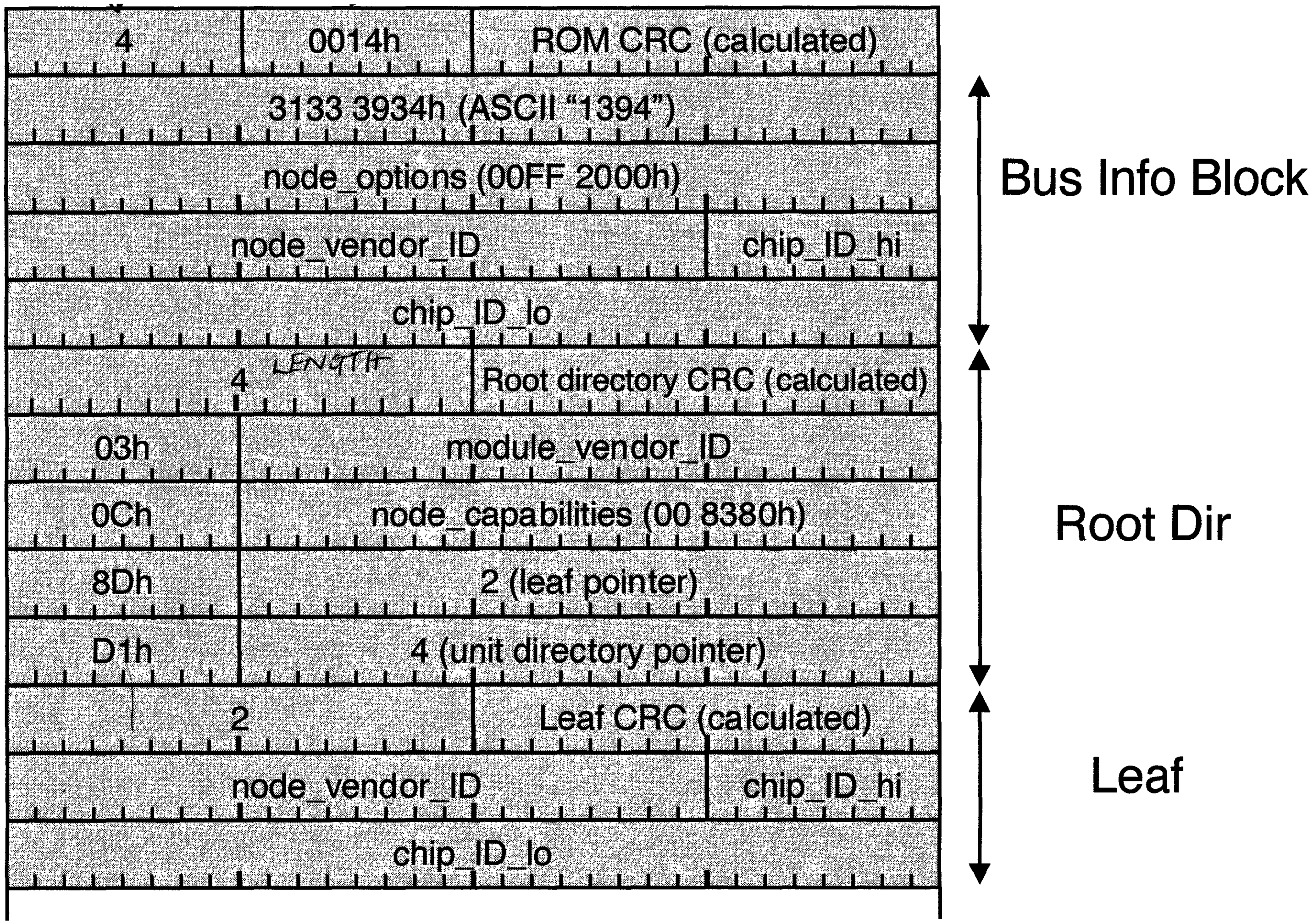
17h-2Fh	reserved for future CSR
30h	Unit power requirements
31h-37h	reserved for bus dependent
38h	Command set spec ID
39h	Command set version
3Ah	Logical Unit characteristics
3Bh-3Fh	allocated by vendors
54h	Management Agent address

Reference - Node Capabilities ROM Entry

0Ch	00h	spt *	ms	int	ext	bas	prv	64 *	fix *	lst *	drq *	r	elo	atn	off	ded	init
8	8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

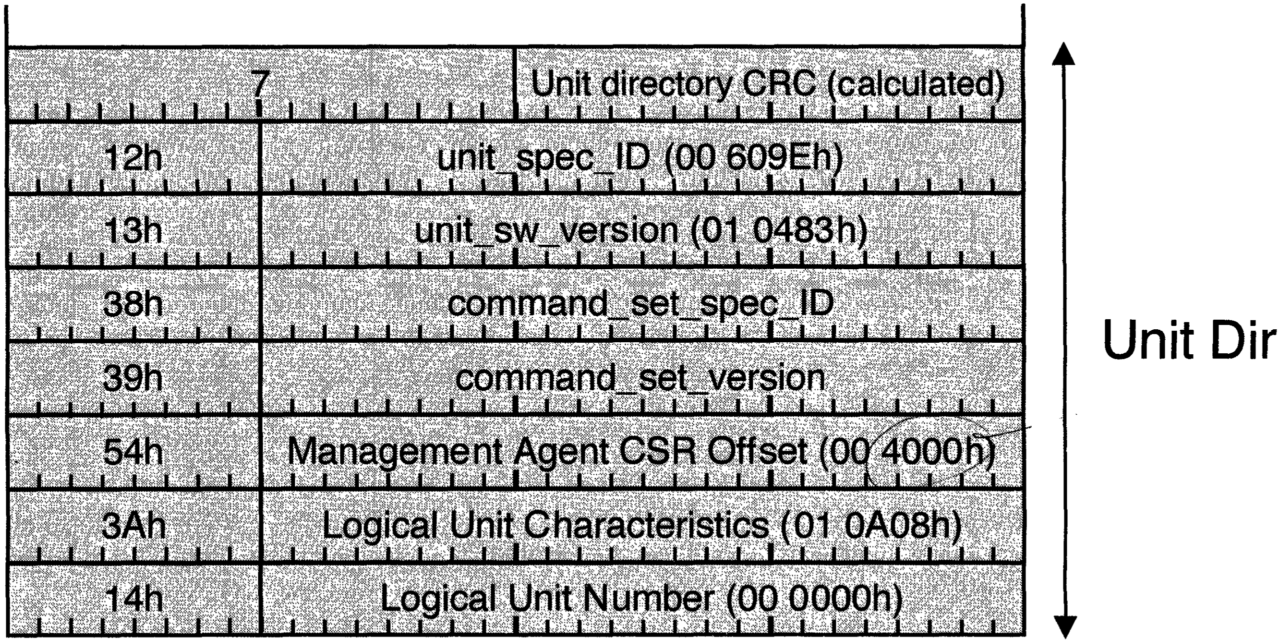
0Ch	Key type & key value
*	Required by 1394-1995
spt	Split timeout implemented
ms	Message passing registers implemented
int	Interrupt target and Interrupt mask registers implemented
ext	Argument registers implemented
bas	Test start and Test State registers implemented
prv	Uses private space
64	Uses 64 bit addressing (otherwise 32 bit addressing)
fix	Uses fixed addressing (otherwise extended addressing)
lst	State Bits.lost implemented
drq	State Bits.dreq implemented (disable requests)
elo	Error log implemented
atn	State Bits.atn implemented
off	State Bits.off implemented
ded	Supports Dead state
init	Supports initializing state

Sample Configuration ROM



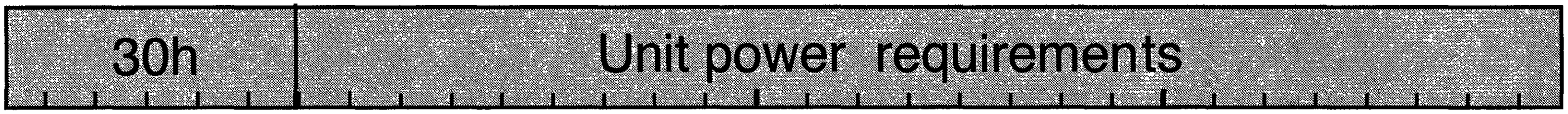
4	0014h	ROM CRC (calculated)	Bus Info Block
3133 3934h (ASCII “1394”)			
node_options (00FF 2000h)			
node_vendor_ID		chip_ID_hi	
chip_ID_lo			
4 LENGTH		Root directory CRC (calculated)	Root Dir
03h	module_vendor_ID		
0Ch	node_capabilities (00 8380h)		
8Dh	2 (leaf pointer)		
D1h	4 (unit directory pointer)		
2		Leaf CRC (calculated)	Leaf
node_vendor_ID		chip_ID_hi	
chip_ID_lo			

Sample Configuration ROM (Continued)



7	Unit directory CRC (calculated)
12h	unit_spec_ID (00 609Eh)
13h	unit_sw_version (01 0483h)
38h	command_set_spec_ID
39h	command_set_version
54h	Management Agent CSR Offset (00 4000h)
3Ah	Logical Unit Characteristics (01 0A08h)
14h	Logical Unit Number (00 0000h)

Reference - Power



30h

Unit power requirements

The 24 bit power requirements field specifies, in deciwatts, the power required by the unit in excess of the power requirements stated in the Self-ID packet. The Self-ID packet will be covered in configuration.

Self powered units will not have this entry in Configuration ROM.

Command Set
Unit Directory Entries

		12	13	38	39
Device Bay		00805F	010000	N/A	N/A
SBP-2 SCSI		00609E	010483	00609E	0104D8
SBP-2 ATA		00609E	010483	00609E	040000
SBP-2 AV/C		00609E	010483	00A02D	010001
Camera	1.04	00A02D	000100	N/A	N/A
	1.20	00A02D	000101	N/A	N/A

12	00609E = NCITS 00A02D = 1394TA
13	010483 = SBP-2
38	Same as 12
39	0104D8 = SCSI 040000 = ATA 010001 = AV/C

Logical Unit Characteristics

3Ah	Reserved	ORB Timeout	ORB Size

ORB Timeout	
ORB Size	In quadlets

Logical Unit Number

14h	r	O	r	Device Type	Logical Unit Number

O	Ordered
r	Reserved
Device Type:	0 = Block Device
	1 = Sequential Device
	2 = Printer
	3 = Processor
	4 = Worm
	5 = CD-ROM
	6 = Scanner
	7 = Optical Memory
	8 = Media Changer
	9 = Communication
	A = Pre-Press
	B = Pre-Press
	C = Enclosure Services
	E = Reduced Block Command

Review

1. How many “state” registers are there in each node?
2. At what address is the “state” register?
3. Which node on a bus has the configuration ROM?
4. Which node has the Root Directory?
5. What is the value after bus reset of the register at offset 21Ch?
6. At what address (64 bits) will I find the beginning of config ROM?

CSRs Notes

CSRs Notes

Section 4

Introduction To SCSI Over 1394

Subjects Covered

Introduction to SCSI

CDB data structure

ORB data structure

SCSI

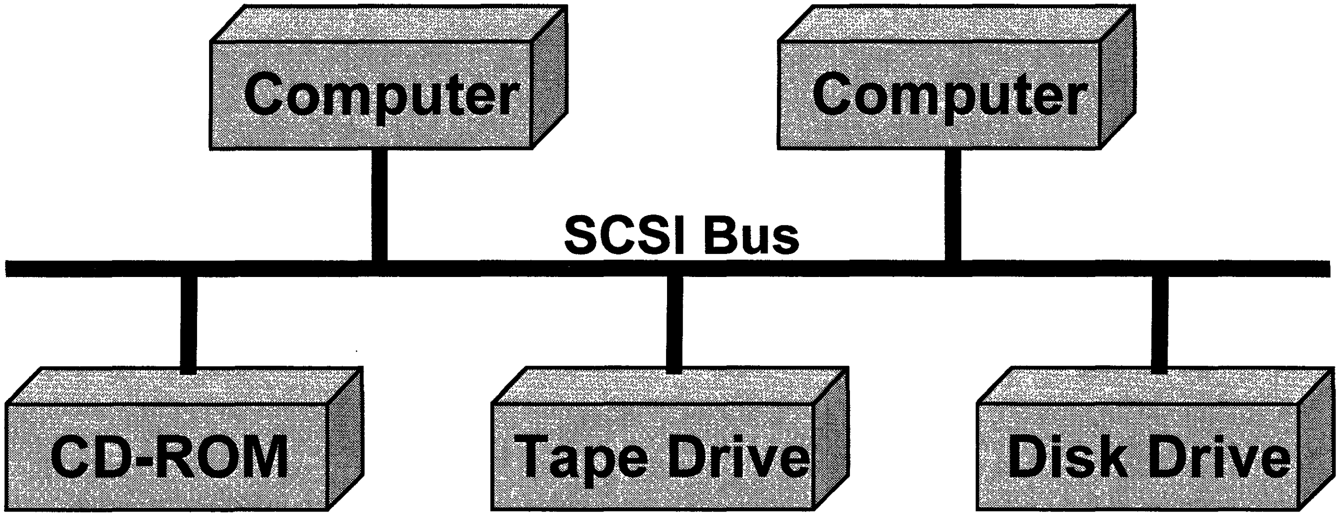
Small Computer System Interface

Specification under the control of the T10 Committee of NCITS
(NCITS = National Committee for Information Technology Standardization, formerly X3 committee of ANSI)

System Level Interface
Drive appears as a 'stack' of logical blocks
Each block has unique Logical Block Address (LBA)

Physical:	50 conductor cable (18 signals), usually ribbon cable (narrow) 68 conductor cable (27 signals), usually ribbon cable (wide)
Logical:	Commands to write/read data to/from LBAs Many, many, many other esoteric commands

SCSI Devices

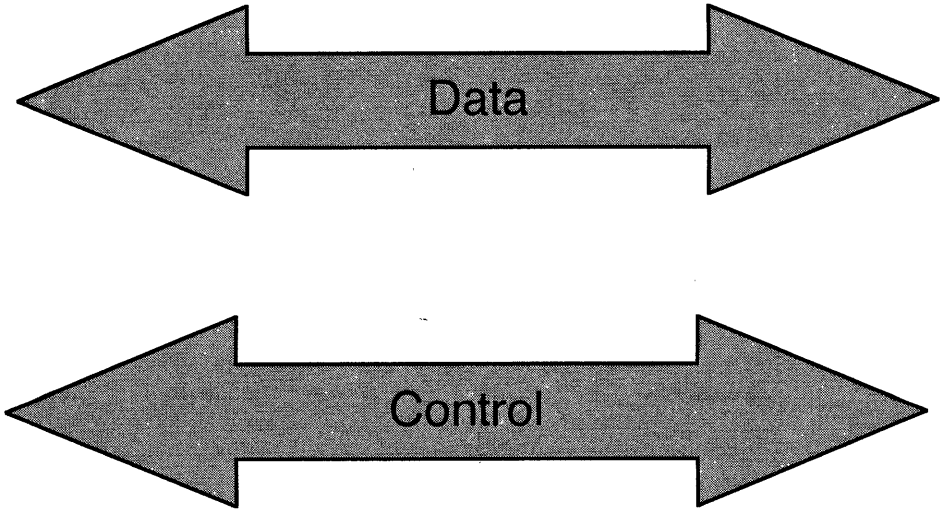


8 or 16 Devices Max

Initiator = Device Originating A Command

Target = Device Responding To A Command

The SCSI Bus



9 Signals To Move Data
(1 Byte + Parity)*

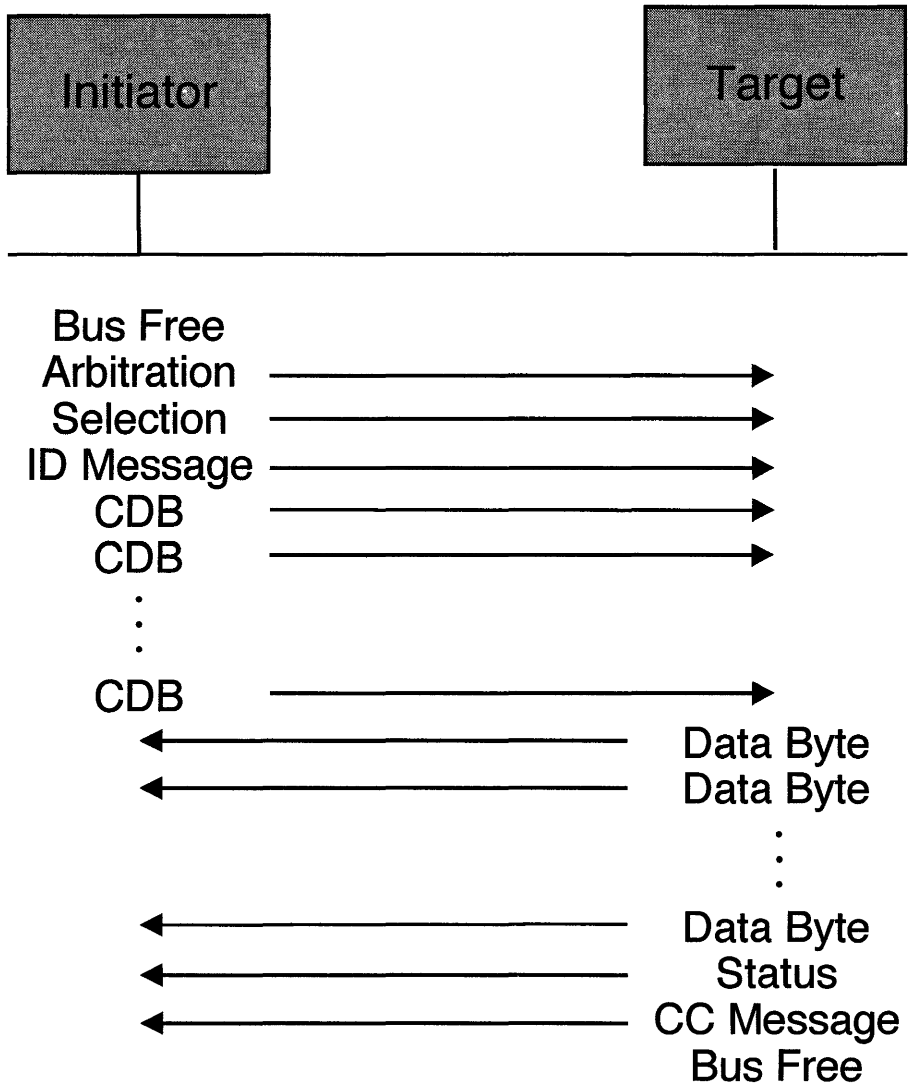
9 Control Signals
Transfer Bytes
Indicates Types of Bytes

Types of Bytes (Phases)

- Arbitration
- Selection
- Command Descriptor Bytes (CDB)
- Data (In/Out)
- Status
- Message (In/Out)

*Wide option with
more signals moves
2 Bytes at a time

Example SCSI Read Command



Initiator

Target

Bus Free
Arbitration →
Selection →
ID Message →
CDB →
CDB →
⋮
CDB →

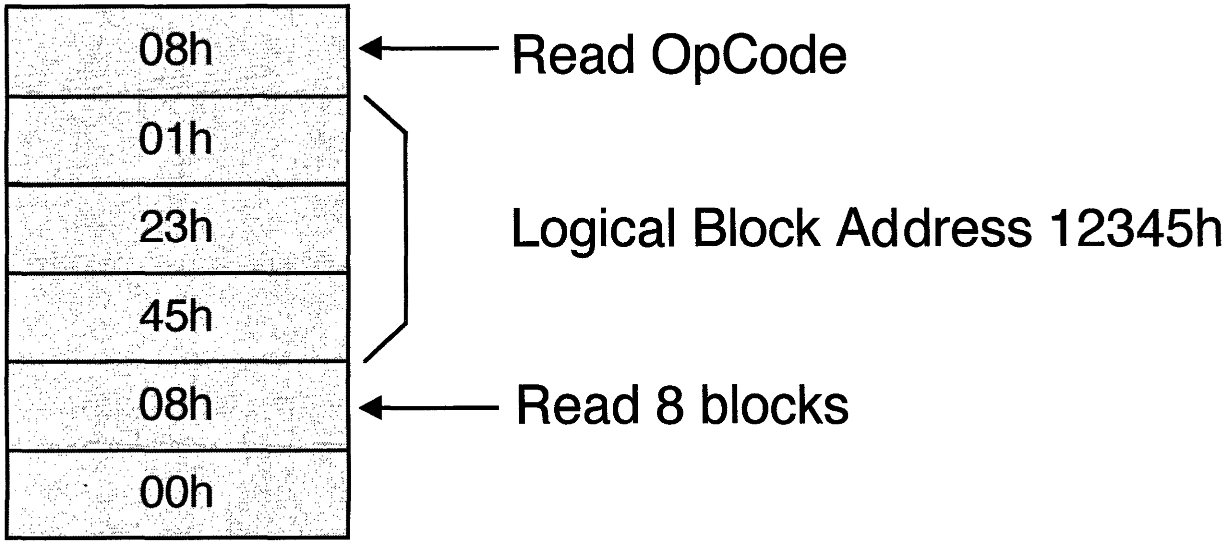
← Data Byte
← Data Byte
⋮
← Data Byte
← Status
← CC Message
Bus Free

Command Formats

6 Byte Command	10 Byte Format	12 Byte Format
OP Code	OP Code	OP Code
LBA$_{20-16}$		
LBA$_{15-8}$	LBA$_{31-24}$	LBA$_{31-24}$
LBA$_{7-0}$	LBA$_{23-16}$	LBA$_{23-16}$
Xfer Length	LBA$_{15-8}$	LBA$_{15-8}$
Control	LBA$_{7-0}$	LBA$_{7-0}$
		Xfer Len$_{31-24}$
	Xfer Len$_{15-8}$	Xfer Len$_{23-16}$
	Xfer Len$_{7-0}$	Xfer Len$_{15-8}$
	Control	Xfer Len$_{7-0}$
		Control

Note: Most commands don't require LBA or Xfer Len and will use these fields differently

Example SCSI Read CDB



08h — Read OpCode

01h
23h
45h

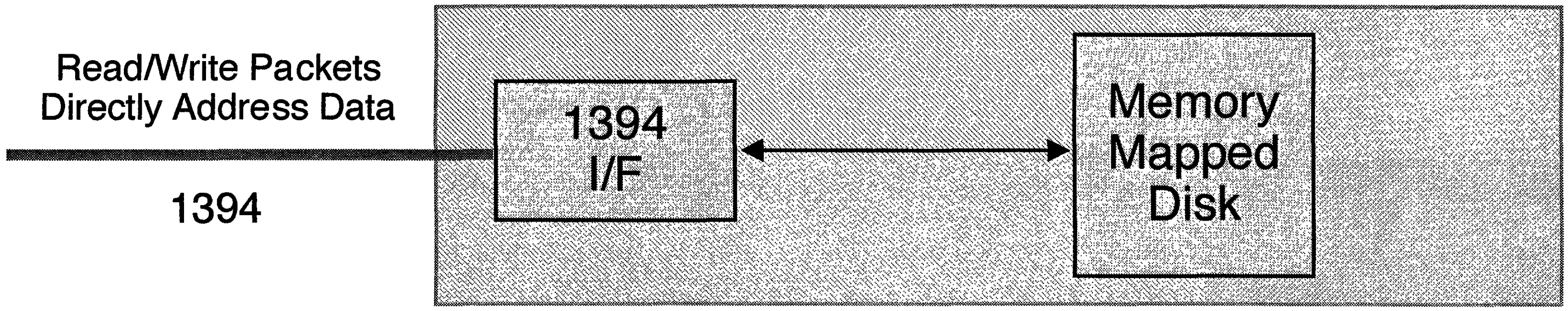
Logical Block Address 12345h

08h — Read 8 blocks

00h

SCSI Over 1394

“Obvious” Way:



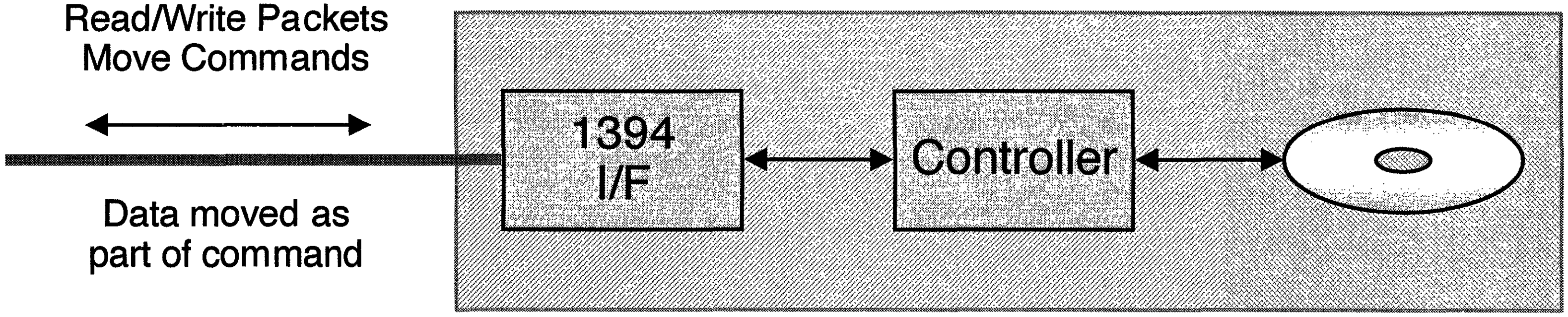
Read/Write Packets
Directly Address Data

1394

1394
I/F

Memory
Mapped
Disk

How it’s done:



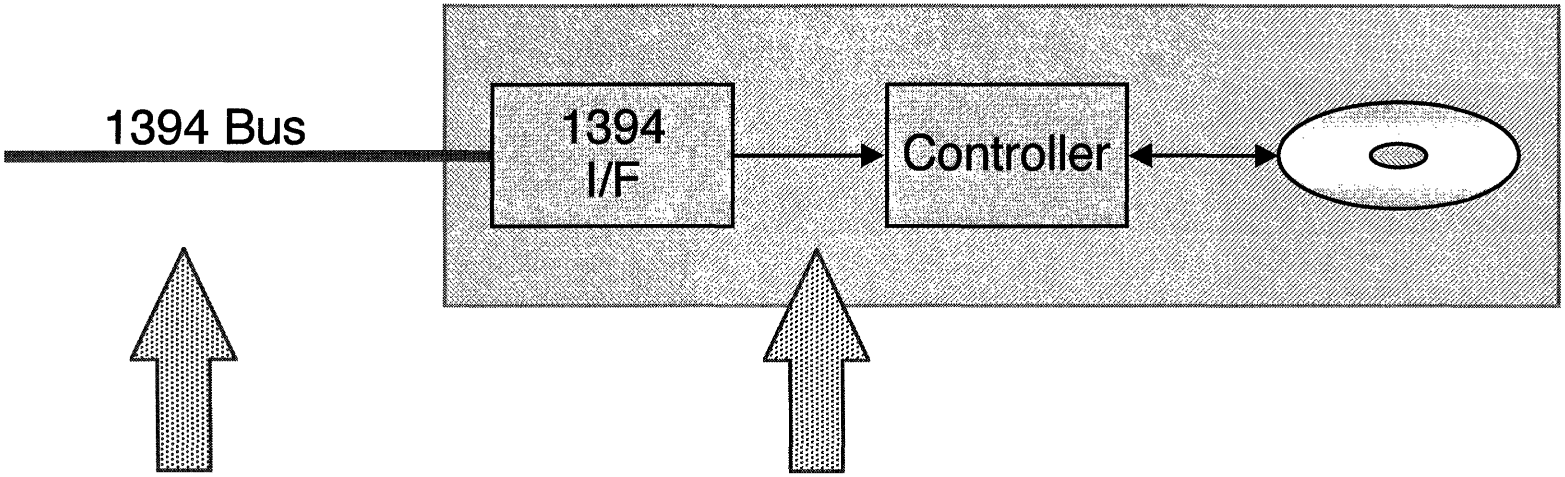
Read/Write Packets
Move Commands

Data moved as
part of command

1394
I/F

Controller

Transactions vs. Commands

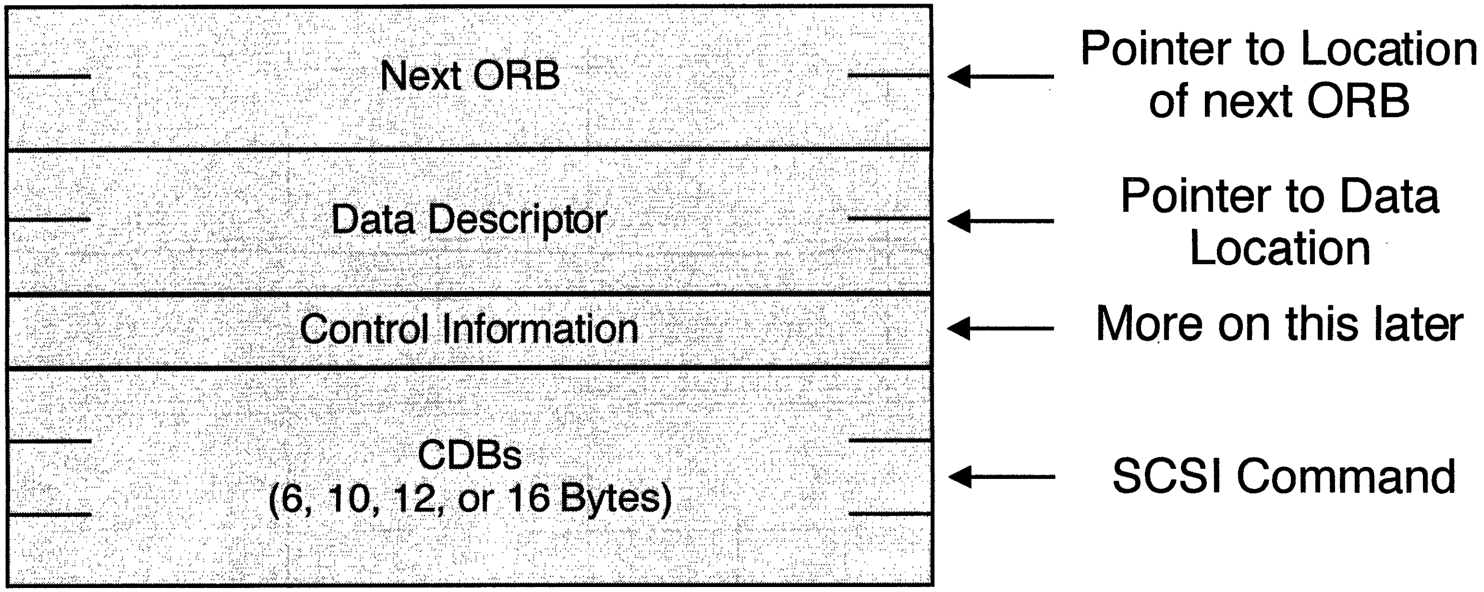


Transactions
Read, Write, & Lock
Issued by any device
(if SCSI could be
initiator or target)

Commands
SCSI Commands
Only Issued by Initiator

ORB - Operation Request Block

SCSI Commands are ‘wrapped” in an ORB:



Next ORB

Data Descriptor

Control Information

CDBs
(6, 10, 12, or 16 Bytes)

Pointer to Location of next ORB

Pointer to Data Location

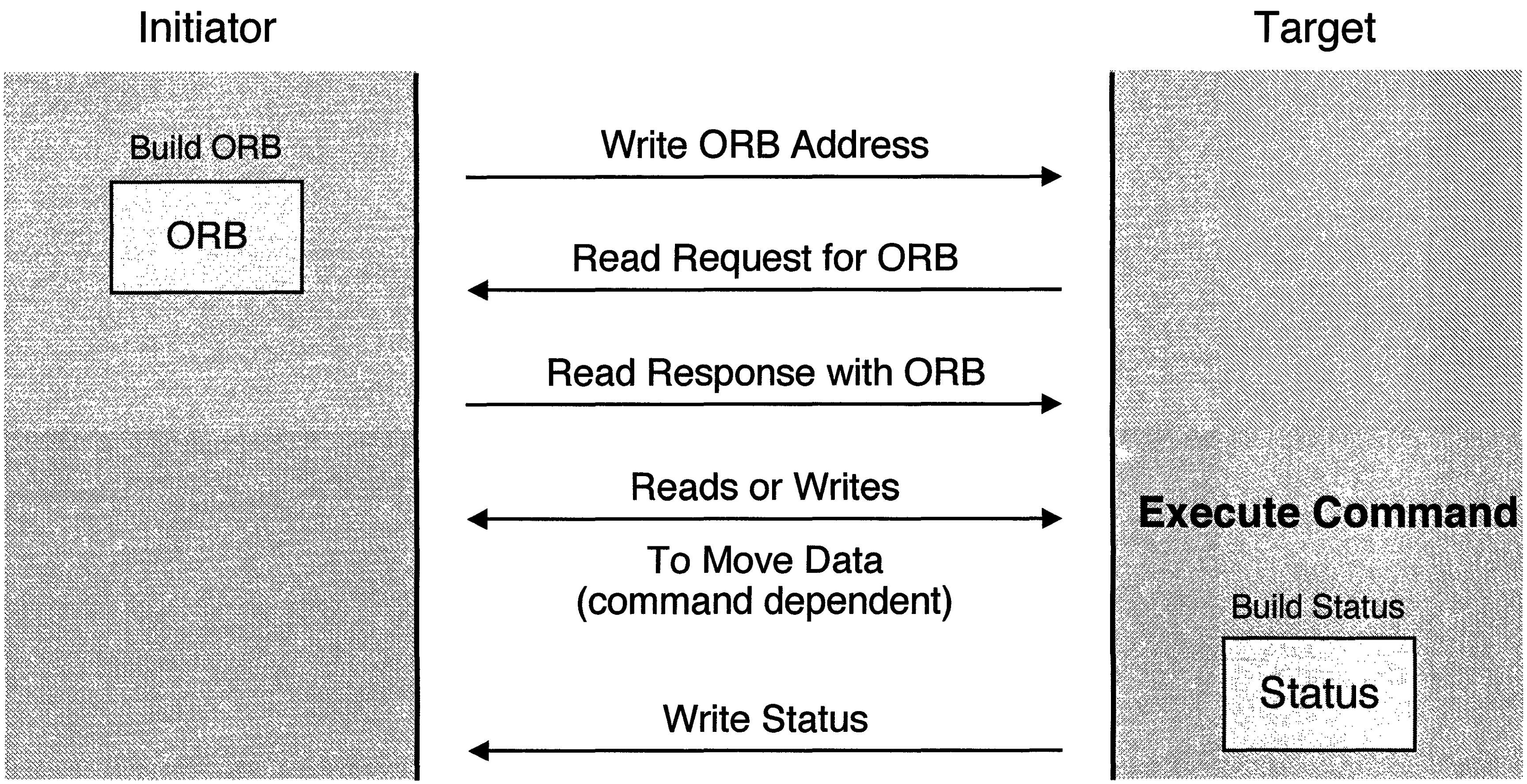
More on this later

SCSI Command

Initiator builds ORB in its memory

Target transfers ORB into the controller

SCSI Over 1394



Initiator

Target

Build ORB

ORB

Write ORB Address

Read Request for ORB

Read Response with ORB

Reads or Writes

To Move Data
(command dependent)

Execute Command

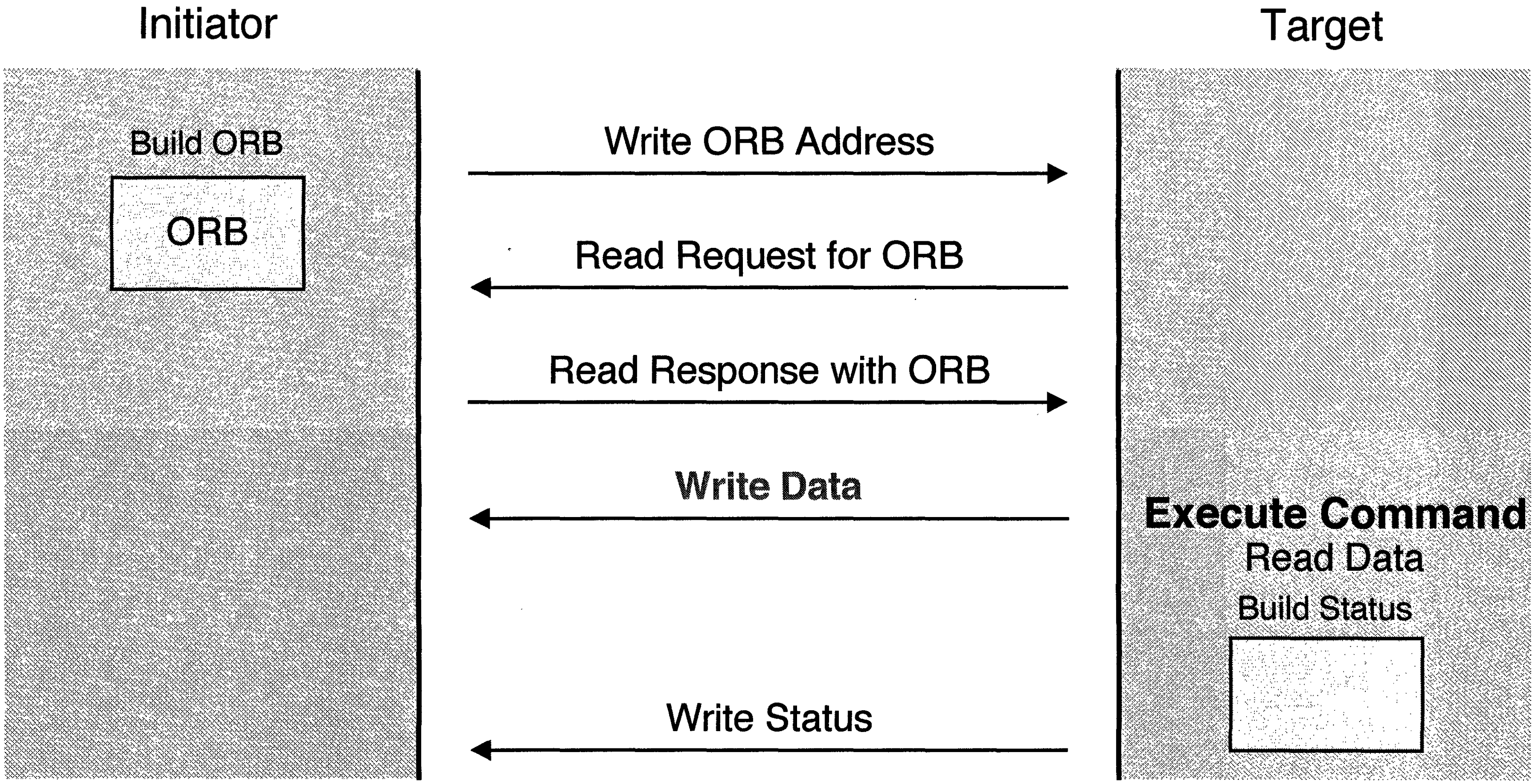
Build Status

Status

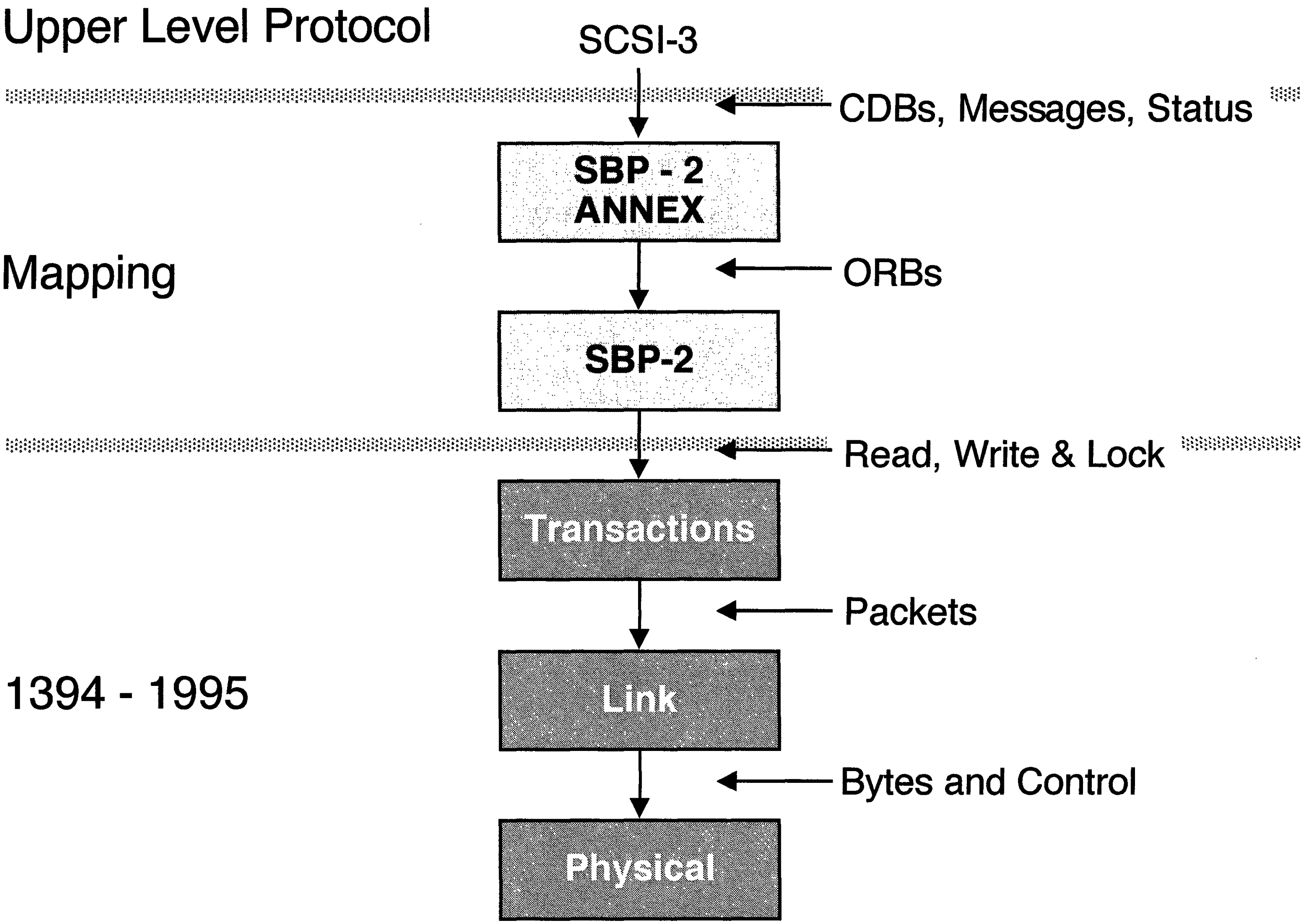
Write Status

Note: ACKs have been omitted to reduce clutter

SCSI Read Processing



Protocol Layers



Upper Level Protocol

SCSI-3

CDBs, Messages, Status

SBP - 2 ANNEX

Mapping

ORBs

SBP-2

Read, Write & Lock

Transactions

Packets

1394 - 1995

Link

Bytes and Control

Physical

Review

1. Differentiate the different levels referenced by the term SCSI
2. What is contained in a CDB?
3. What do the acronyms CDB and ORB mean?
4. What is contained in an ORB?

SCSI Over 1394 Notes

Section 5

Serial Bus Protocol SBP-2

Subjects Covered

Command ORBs and fields

Management ORBs and fields

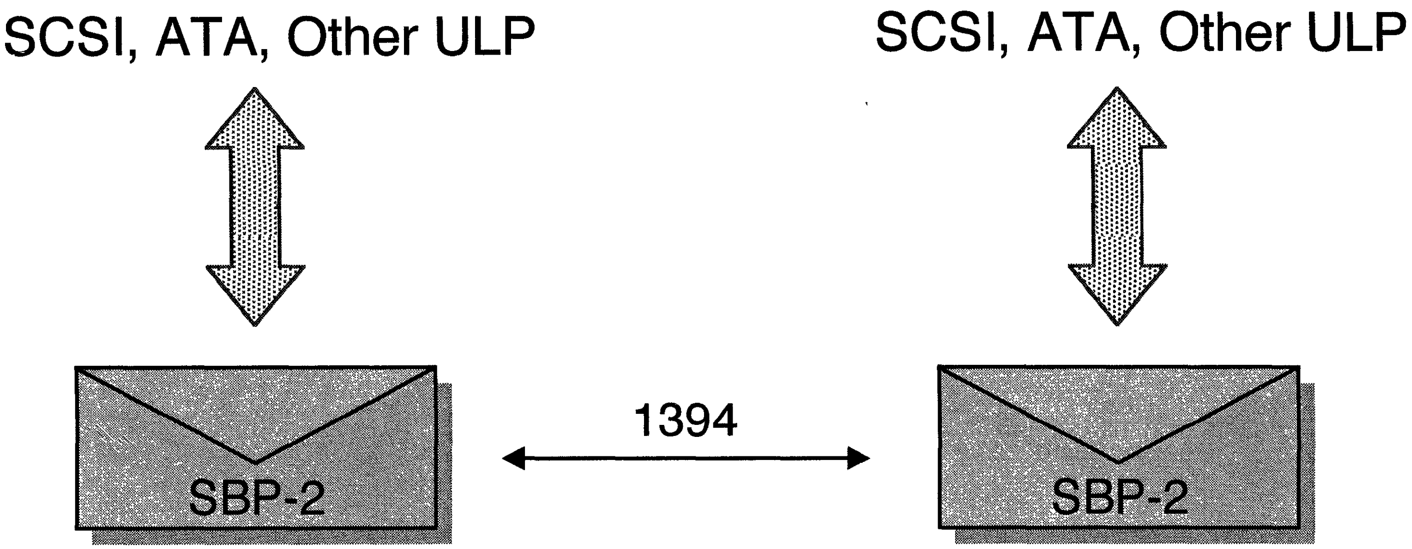
Status Blocks

Login and Resets

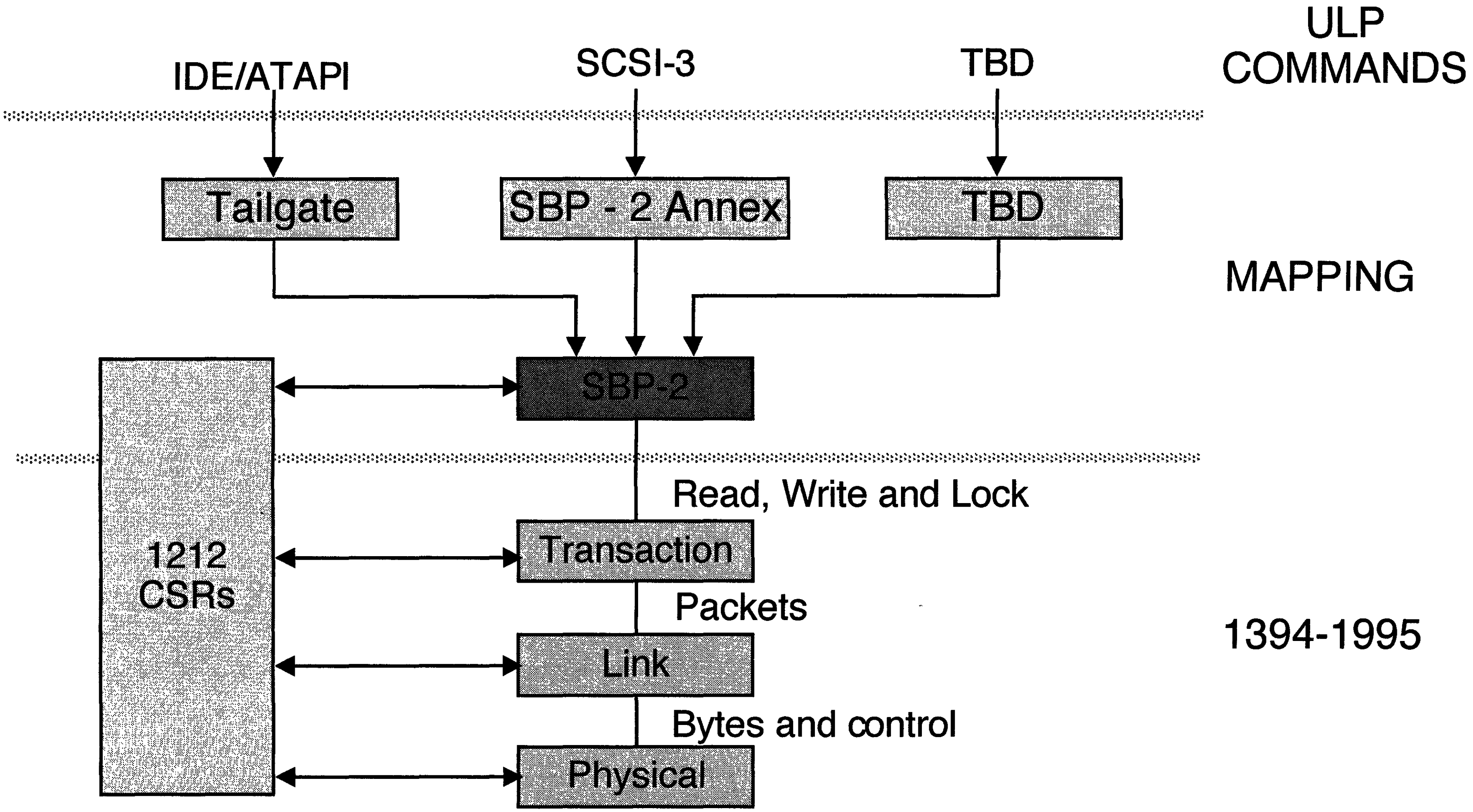
Serial Bus Protocol: SBP-2

Maps Upper Level Protocols (ULPs) onto 1394

ULPs = SCSI, ATA, IP, ??



Protocol Layers



© KnowledgeTek, Inc.
rev 5.41d 15.H.08

SBP-2 Overview

Command set neutral

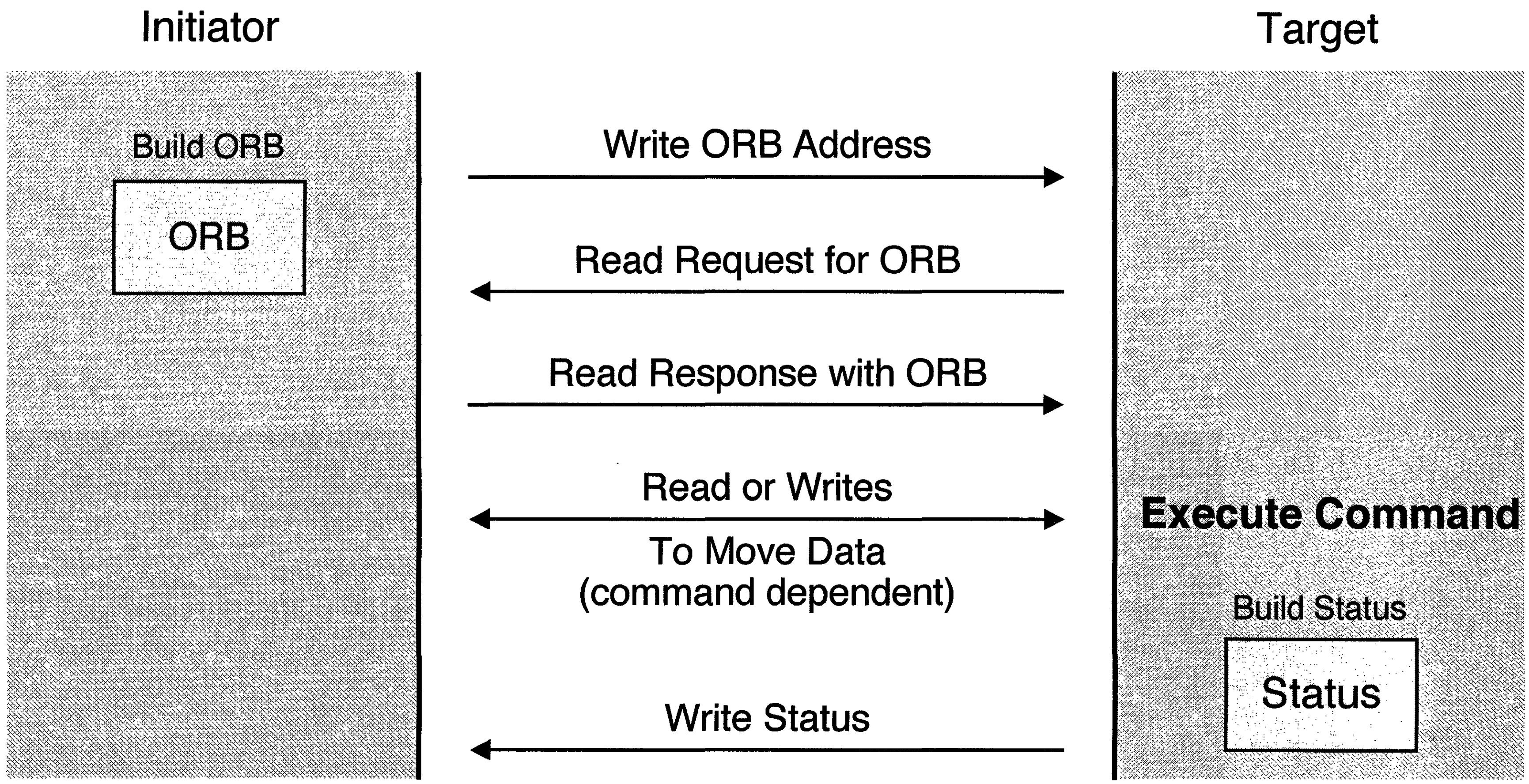
Current plans address SCSI, ATA, ATAPI, IP

ULP Commands are packaged in an ORB (Operation Request Block)

Devices must login before sending commands

- Exchange certain operational information

SBP - 2 Command Process



Initiator

Target

Build ORB

ORB

Write ORB Address

Read Request for ORB

Read Response with ORB

Read or Writes

To Move Data
(command dependent)

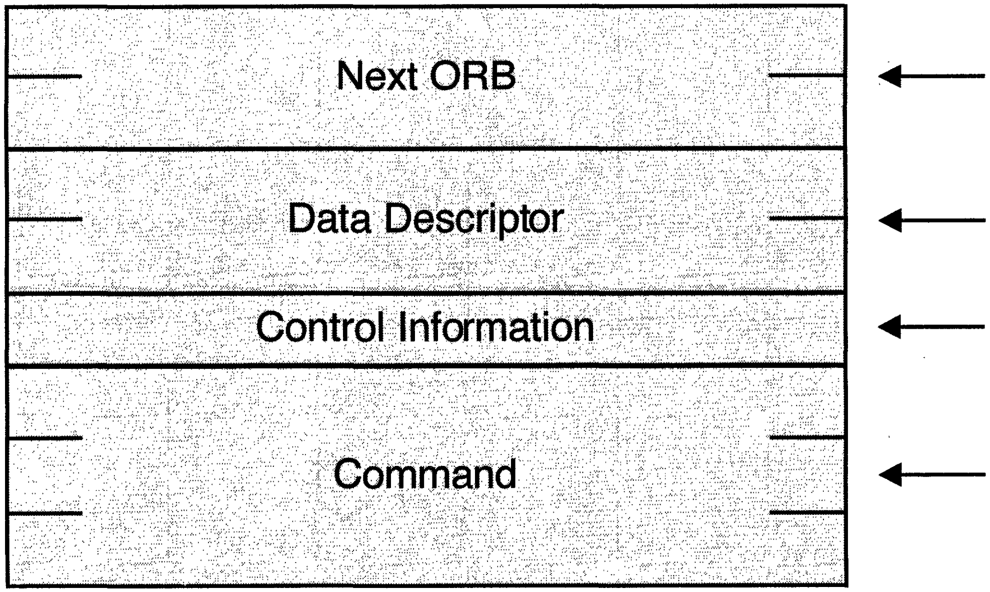
Write Status

Execute Command

Build Status

Status

Operation Request Block - ORB



Next ORB

Data Descriptor

Control Information

Command

Pointer to Next ORB
Allows chaining commands
Set to Null if no more ORBs in chain

Pointer to Data Buffer

Info on how to execute command
at the 1394 Bus Level

Upper Level Protocol (ULP) command
SCSI CDBs
ATA Command Block Registers
ATAPI Packet Command
TBI*

* To be invented

ORB Control Information

N	Req Fmt	r	D	spd	Max Payload	P	PSize	Data Size

N — Notify - Initiator Status has been posted
1= Post status at end of this ORB
0= Don't post status unless there's an error

Req Fmt — Request Format
0= SBP-2, 1= Reserved, 2= Vendor dependent, 3= Dummy ORB

D — Direction: 0 = Data transfer into target memory
1= Data transfer into initiator memory

Speed — 0= S100, 1= S200, 2= S400, 3= S800, 4= S1600, 5= S3200

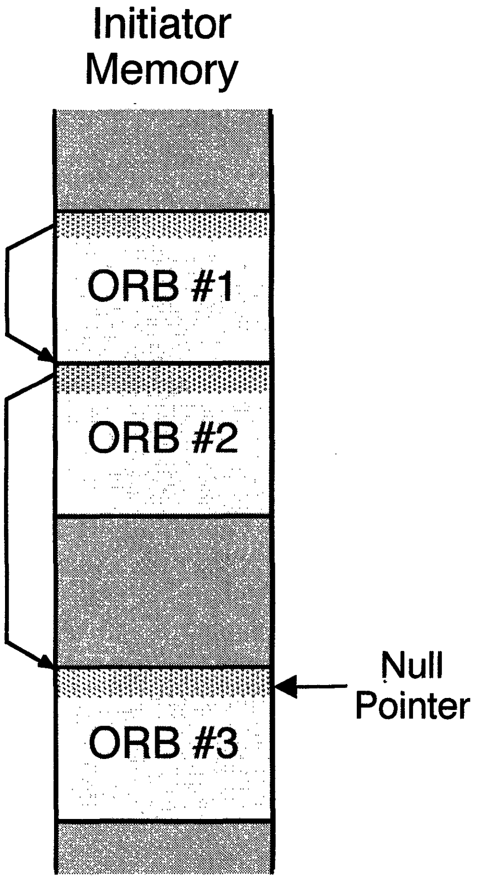
Max Payload — Maximum number of bytes in a single read or write = $2^{(\text{max pay} +2)}$

P — Page table present - Indicates Data Descriptor uses Indirect Mode

PSize — Determines size of the pages for Indirect Data Descriptors

Data Size — Size in bytes of the system memory of the Data Buffer (P=0)
Number of elements in Page table (P=1)

Using The Next ORB Pointer



Initiator builds Linked-List of ORBs

Writes address of first to Target

Target reads ORB #1

Executes ORB #1

Writes Status for ORB #1

Uses Next ORB Pointer to read ORB #2

Executes ORB #2

Writes Status for ORB #2

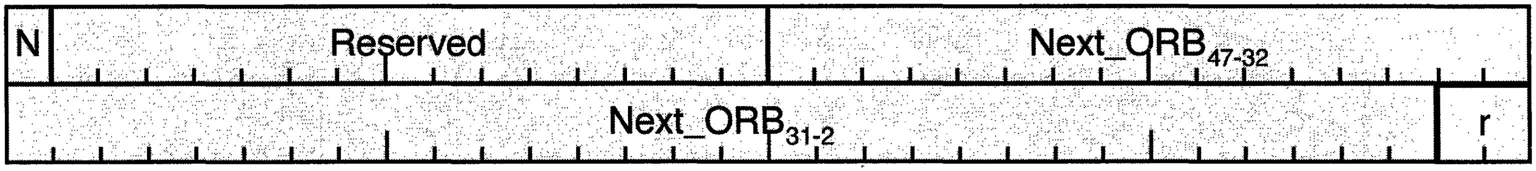
Uses Next ORB Pointer to read ORB #3

Executes ORB #3

Writes Status for ORB #3

Next ORB Pointer = Null Indicates done

Next ORB Pointer Format



N Null Flag Bit

0 = Offset hi and Offset lo are the address for the next ORB

1 = This is the last ORB in the link list, ignore offset hi and offset lo

Reserved Set to Zero

Next_ORB Address of Next ORB in Initiators Node Memory Space

ORB must start on Quadlet boundary

(Bottom two bits of address must be zero)

Data Descriptor

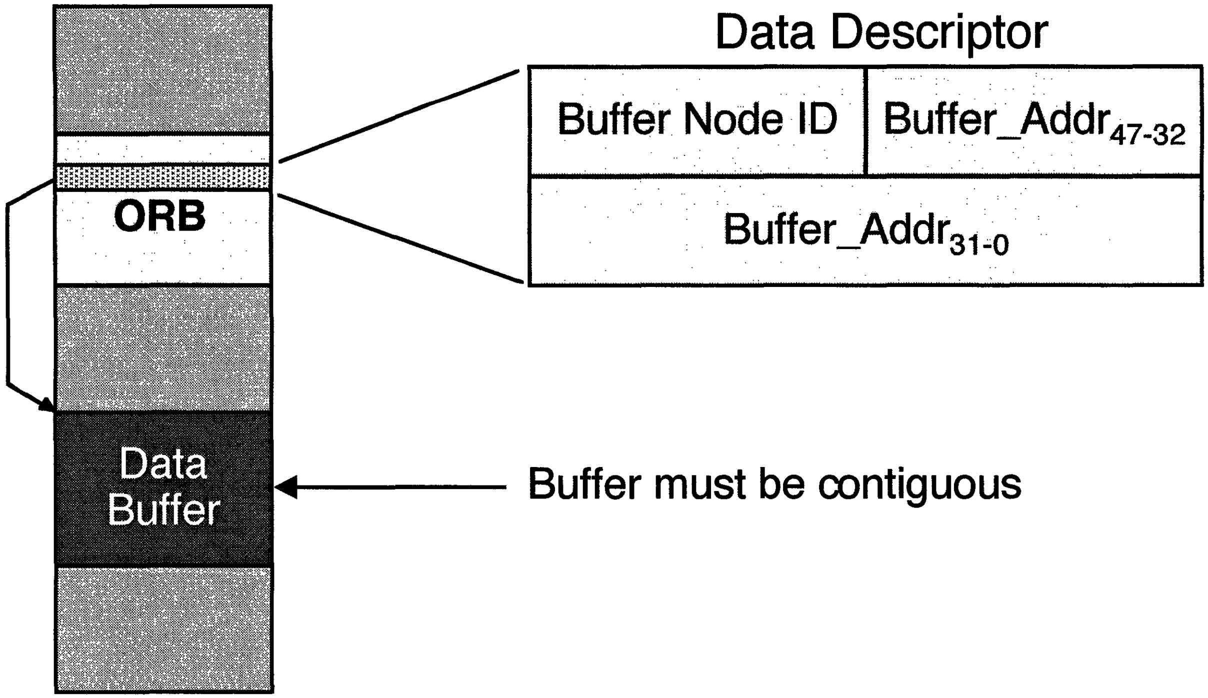
Location of the Data Buffer

- Read Commands - Data placed here
- Write Commands - Data taken from here

Two Modes of Operation

- Direct - Data Descriptor contains address
- Indirect - Data Descriptor contains address of Page Table
- P flag in Control Info indicates which

Using The Data Descriptor In Direct Mode
(P Flag = 0)

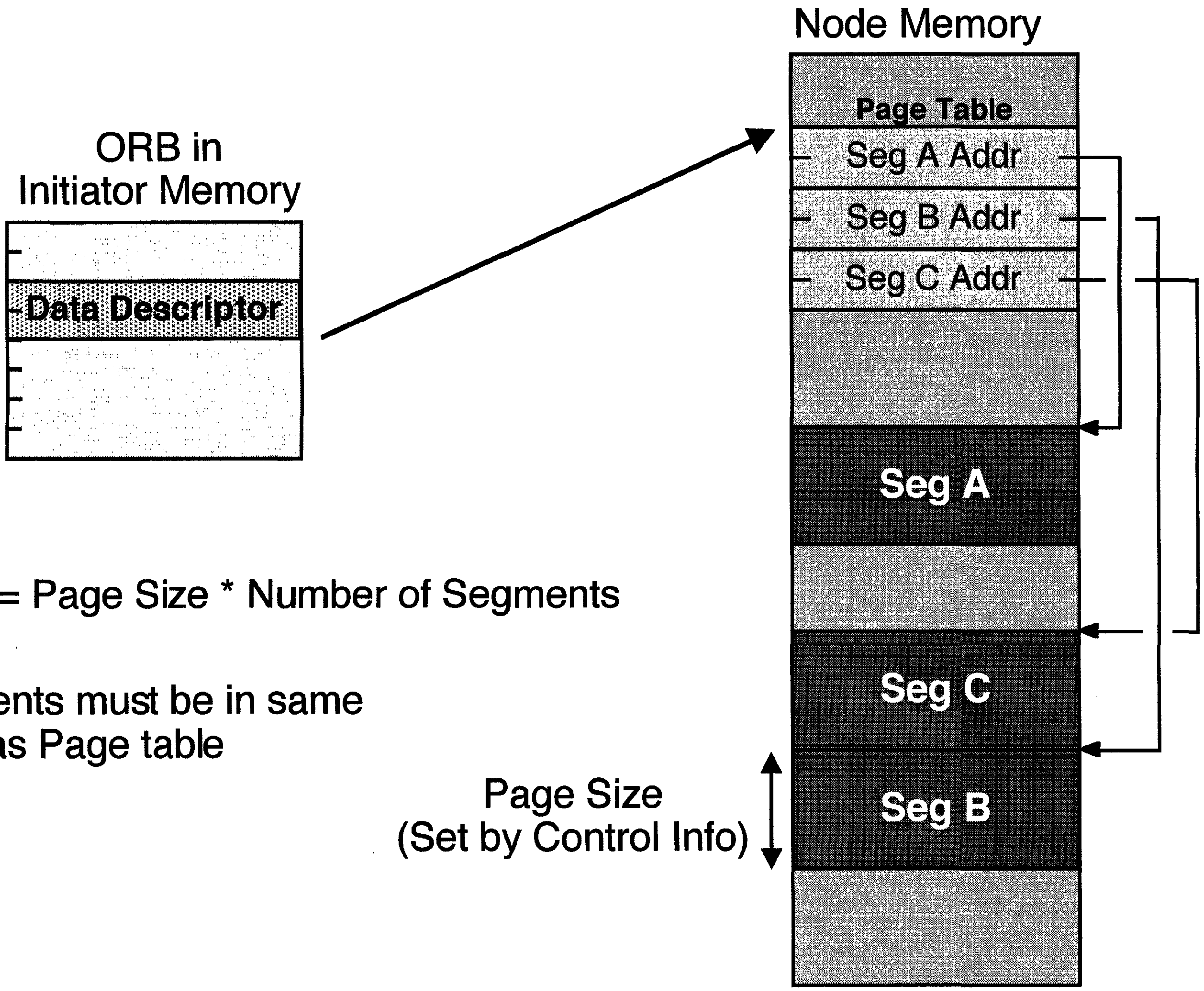


Buffer must be contiguous

Note: Data Buffer can be located **anywhere** not just in Initiator's Node!

Using The Data Descriptor In Indirect Mode

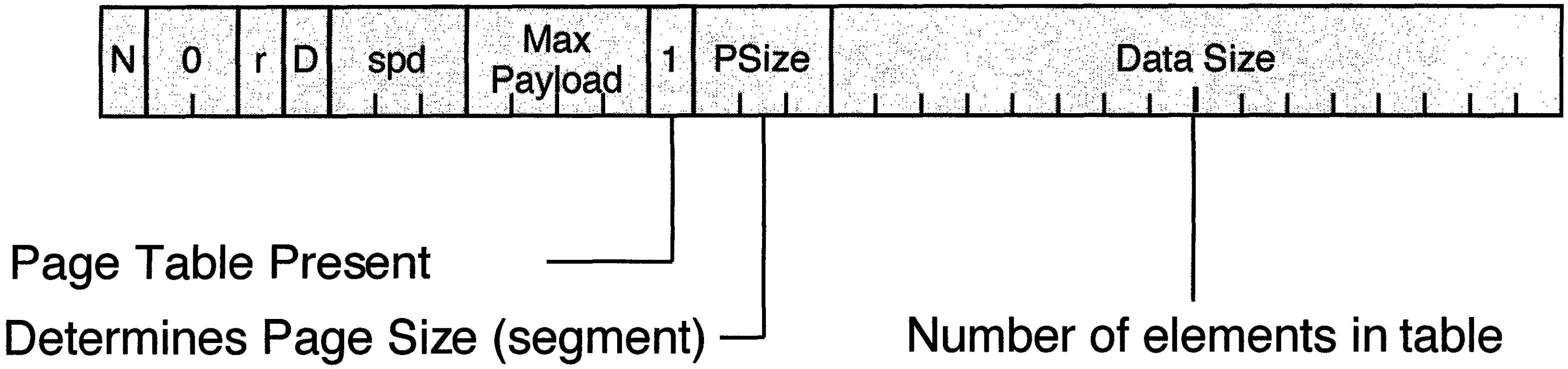
(P Flag = 1)



Data Length = Page Size * Number of Segments

Note: Segments must be in same node as Page table

Control Info For Indirect Data Descriptors



N	0	r	D	spd	Max Payload	1	PSize	Data Size

Page Table Present

Determines Page Size (segment)

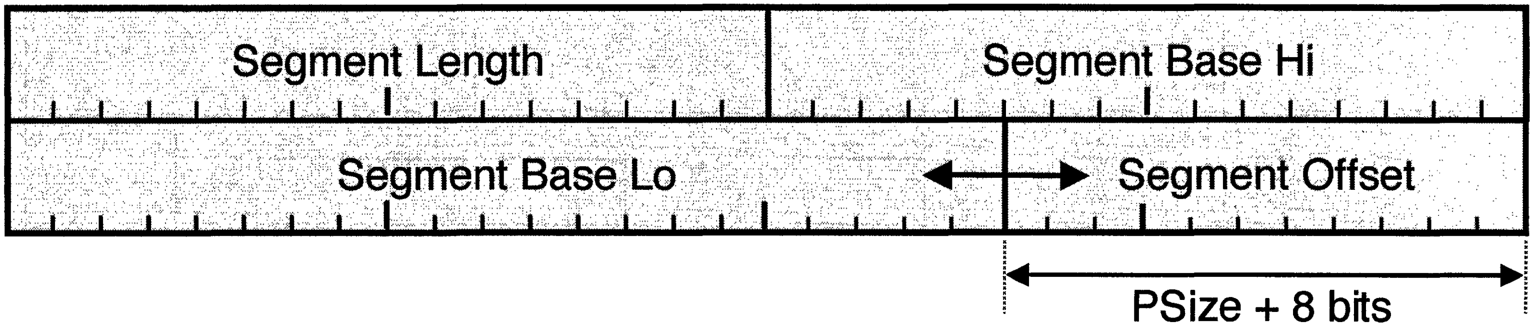
Number of elements in table

$$\text{Page Size} = 2^{(\text{PSize} + 8)}$$

PSize =	Page Size
0 0 0 =	256 Bytes
0 0 1 =	512 Bytes
0 1 0 =	1K Bytes
0 1 1 =	2K Bytes
1 0 0 =	4K Bytes
1 0 1 =	8K Bytes
1 1 0 =	16K Bytes
1 1 1 =	32K Bytes

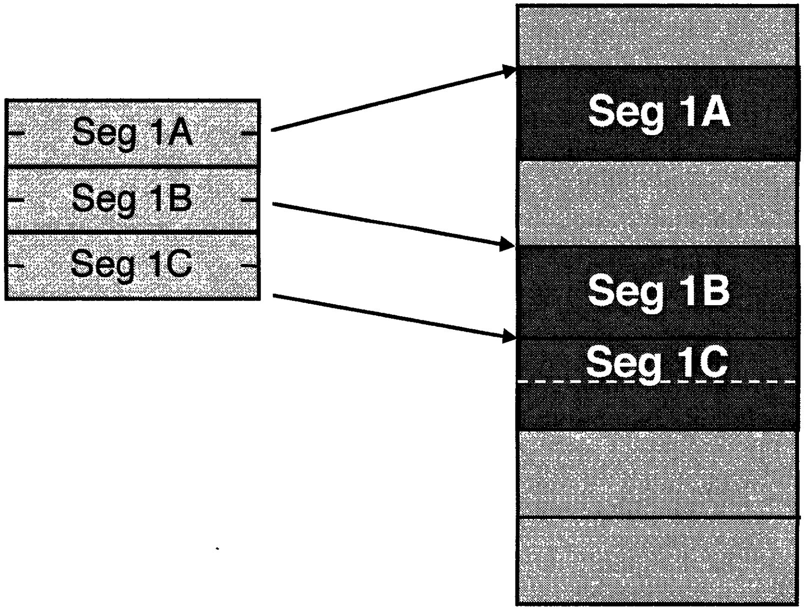
Page Table Format

Page Table composed of Page Table Elements



Segment Length	Number of Bytes used in this segment Normally equal to Page Size
Segment Base	Address of 1st Byte of Segment Node ID same as for Page Table Append (PSize + 8) zero bits
Segment Offset	Used in Offset Transfers Must be zero in all but 1st Page Table Element

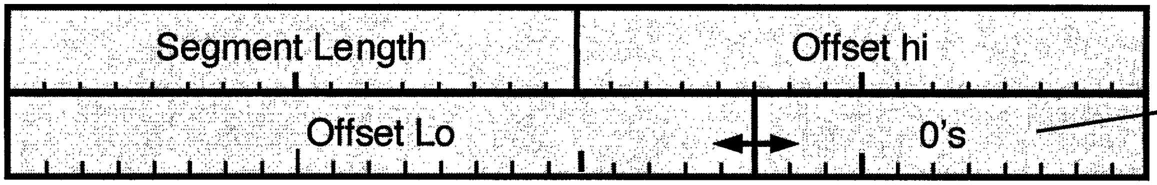
Normalized Page Tables



Segment Length = Page Size

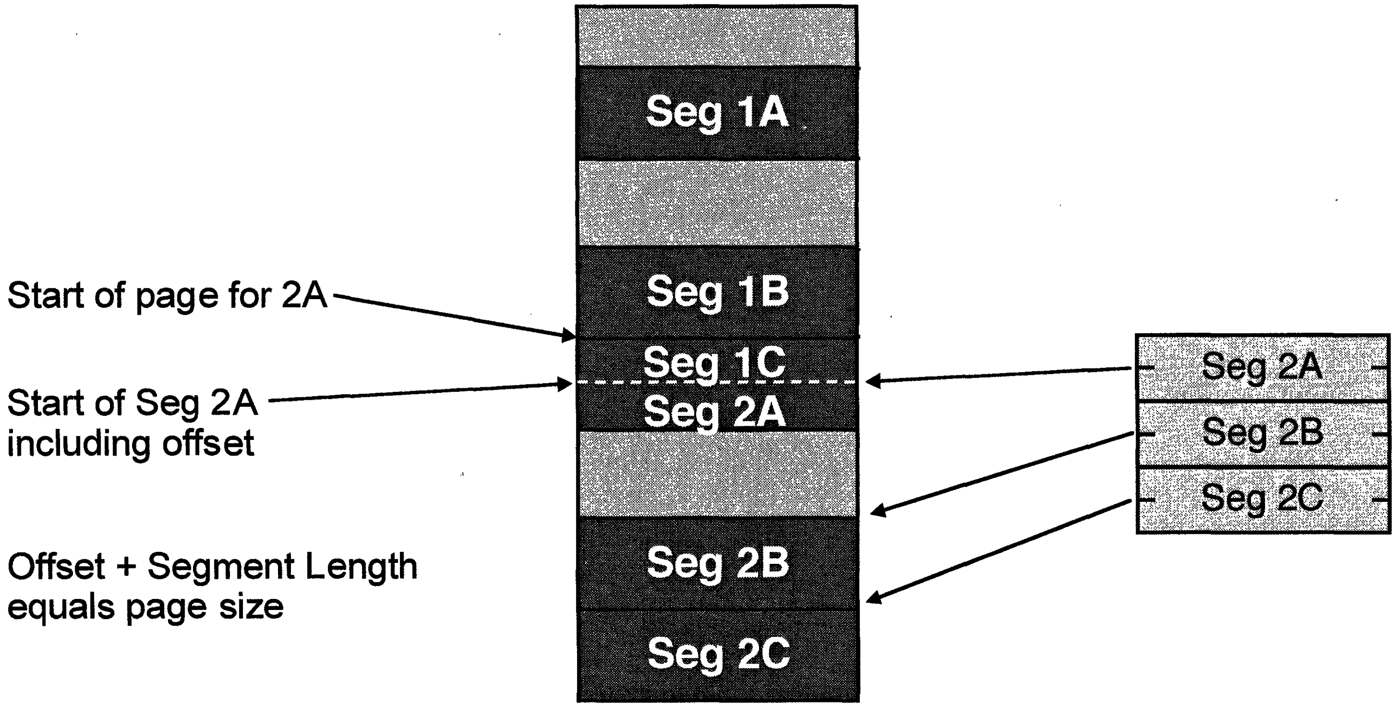
Segment Length = Page Size

Segment Length = Remaining Data



for page alignment

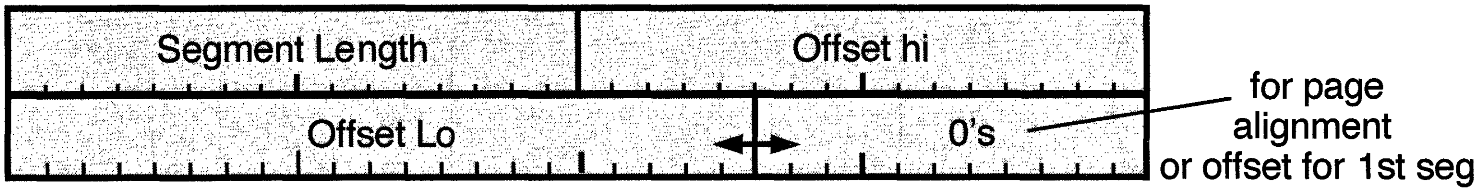
Normalized Page Tables



Start of page for 2A

Start of Seg 2A including offset

Offset + Segment Length equals page size



Segment Length	Offset hi
Offset Lo	0's

for page alignment or offset for 1st seg

Status Block

SRC	Resp	D	Len	SBP Status	ORB Offset Hi
ORB Offset Lo					
Command Set Dependent					

Note: If there is no error, the target need only post the first two quadlets of status

Status Block Definitions

ORB Offset — Identifies ORB for this status

Resp — Response

- 0 = Request complete. The request completed without transport protocol error.
- 1 = Transport failure. Target detected nonrecoverable transport error.
- 2 = Illegal request. Unsupported bit or field in ORB.
- 3 = Vendor dependent.

D — 1 = Target transitioned to dead state

Len — Length. Number of valid quadlets -1 stored as status (Value of 7 means 8 quadlets were stored)

SRC

- 00b Solicited Status, not end of list
- 01b Solicited Status, next ORB = Null — status in response to ORB
- 10b Unsolicited Status
- 11b Reserved

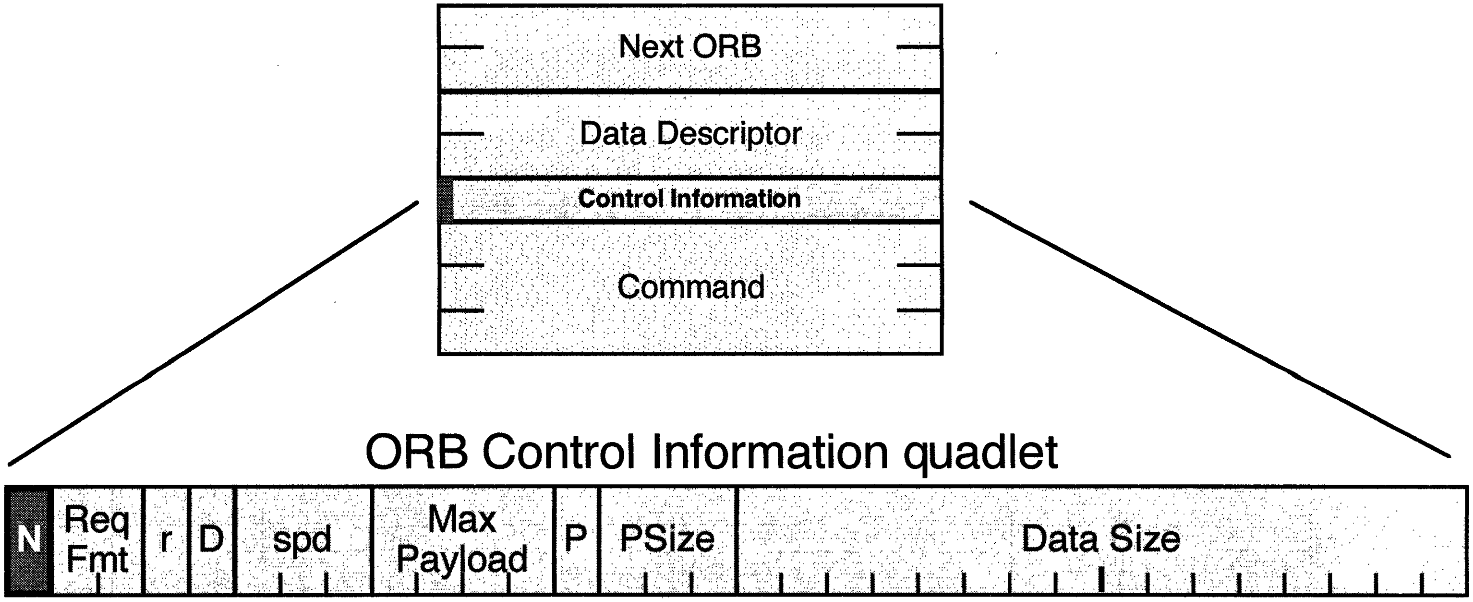
V

SBP-2 Status Block Definitions

The following are valid only if Resp = 0, Request Complete

0	=	No additional sense to report
1	=	Invalid request type
2	=	Speed not supported
3	=	Page size not supported
4	=	Access denied
5	=	Logical unit not supported
6	=	Maximum payload too small
7	=	Too many channels
8	=	Resources unavailable
9	=	Function rejected
A	=	Login ID not recognized
B	=	Dummy ORB completed
C	=	Request aborted
FF	=	unspecified error

Notify Bit



Next ORB

Data Descriptor

Control Information

Command

ORB Control Information quadlet

N	Req Fmt	r	D	spd	Max Payload	P	PSize	Data Size

N Notify Initiator that status has been posted.

1 - Post status at the completion of this ORB

0 - Only post status if it terminated in an error

Review

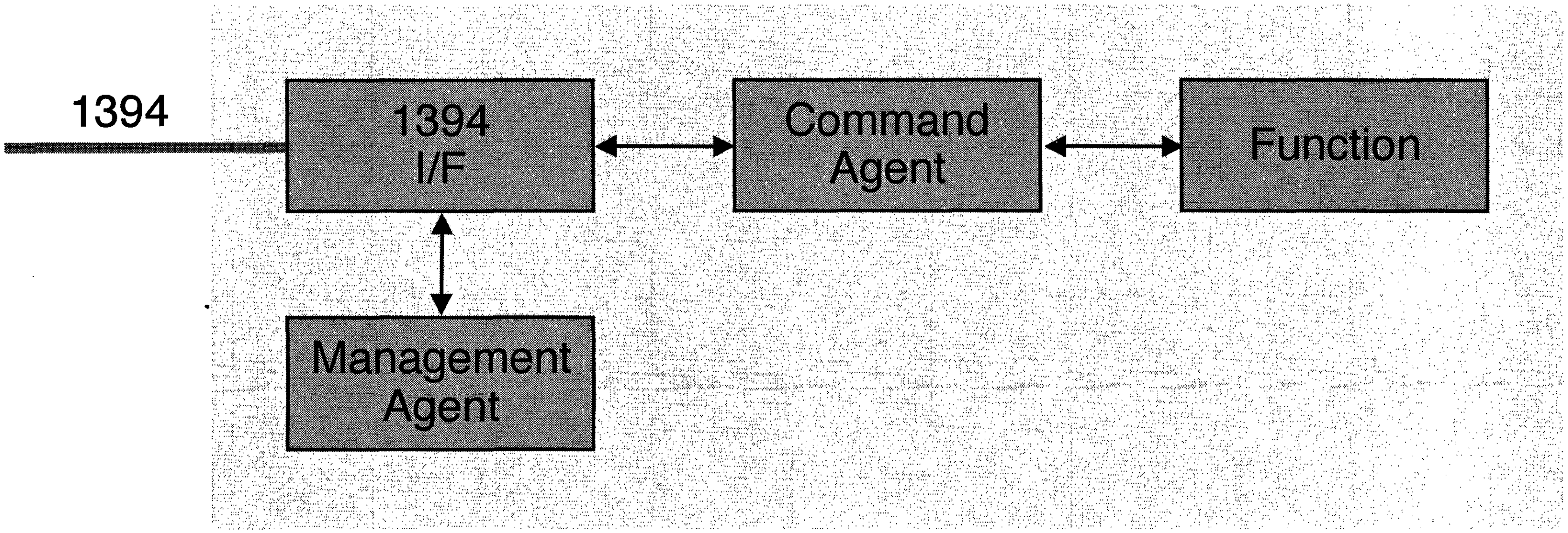
Covered

- Initiator built Command ORB and notified target
  - SCSI commands are covered in the section 6
  - ATA commands are covered in the section 7
- Target fetched ORB with a read transaction
- Target executed command
- Target used write transaction to send status to Initiator

Yet to cover

- How does Initiator know where to write ORB address?
- How does Target know where to write status?
- How does the Target control who is sending it commands?
- Can the Initiator send several commands at once (Stream)?
- Management vs. Command ORBs

Agents



Management Agent - Manages Node

Login, Logout, Reset, etc.

Command Agent - Performs the Command

(Device Controller) stuff w/ ULP

Management Vs. Command ORBs

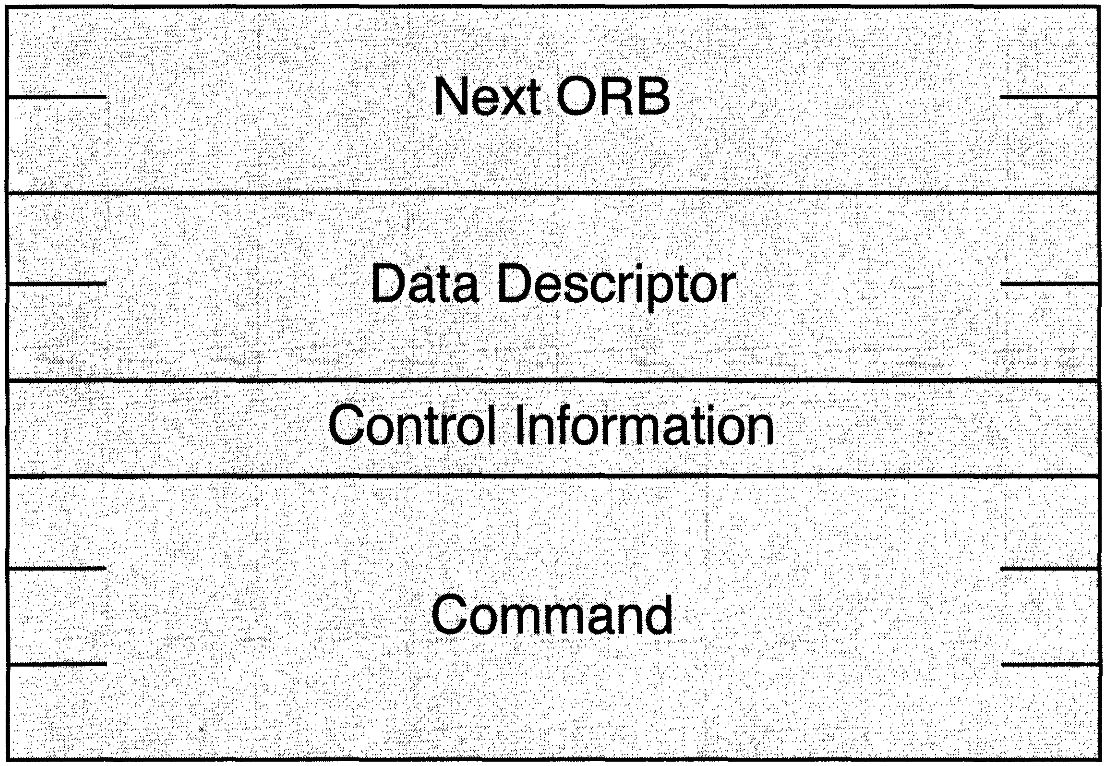
Management ORBs

- Sent to the Management Agent
- Execute a single function only (can't be linked)
- Functions aimed at the node
  - Login, Logout, Logical Unit Reset, Target Reset, etc.

Command ORBs

- Sent to the Command Agent
- May be connected in Link Lists
- Functions aimed at the device (ULP Commands)

Command ORB Format



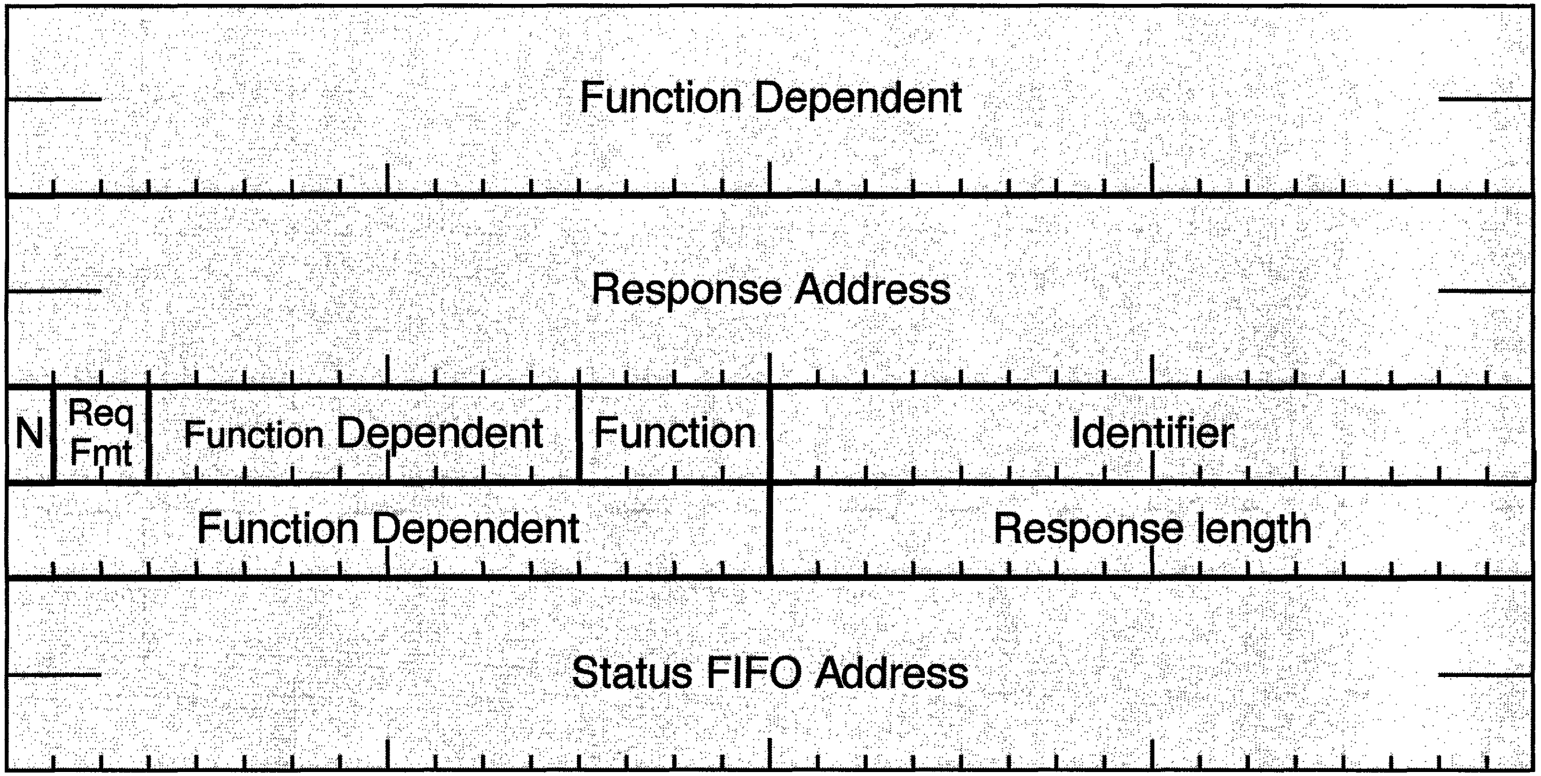
Next ORB

Data Descriptor

Control Information

Command

Management ORB Format



Function Dependent

Response Address

N	Req Fmt	Function Dependent	Function	Identifier

Function Dependent	Response length

Status FIFO Address

Note: All currently defined Management ORBs adhere to this format. However, the SBP-2 standard does not specify that future Management ORBs will necessarily follow this format. The standard specifies the format on a function by function basis.

Management ORB Fields

Response Address — Location in system memory to write the response to this ORB

N — Notify Status Flag
1 = Always report Status
0 = Only report Status on Errors

Req Fmt — Set to 00

Response Length — Space reserved for Response at Response Address

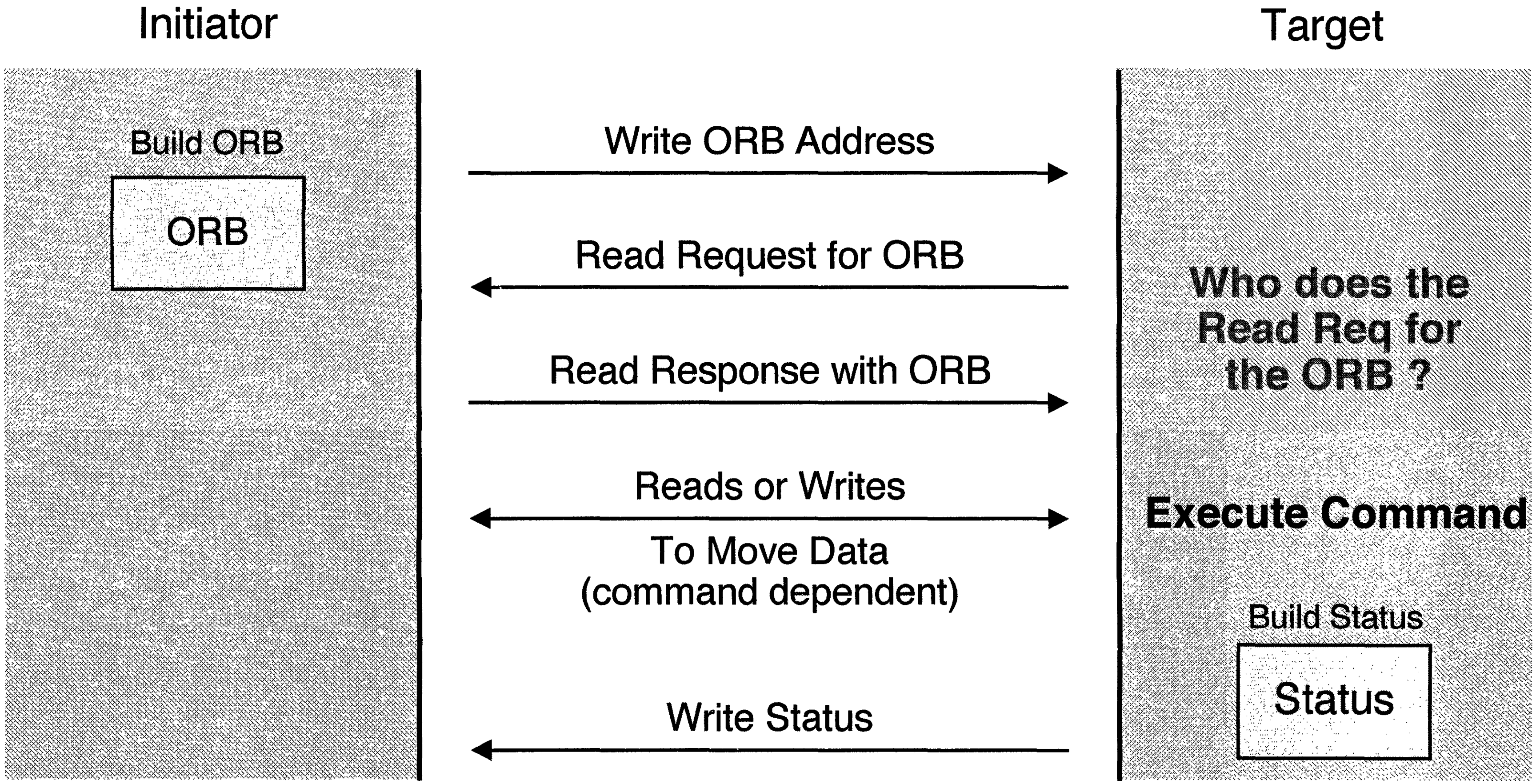
Status FIFO — Location in system memory to write the status block for this ORB

Identifier — Identifies who the ORB is for
LUN on Logins
Login ID on the other Management ORBs

Management Functions

Value	Management Function
0	Login
1	Query Logins
2	Reserved
3	Reconnect
4	Set Password
5-6	Reserved
7	Logout
8-A	Reserved
B	Abort Task
C	Abort Task Set
D	Reserved
E	Logical Unit Reset
F	Target Reset

The SBP - 2 Command Process



How does the initiator know where to write the ORB address ?

How does the target know where to write the status ?

Login

Management ORB to Management Agent

Performed before any Command ORBs are sent by Initiator

Tells Target where to return Status

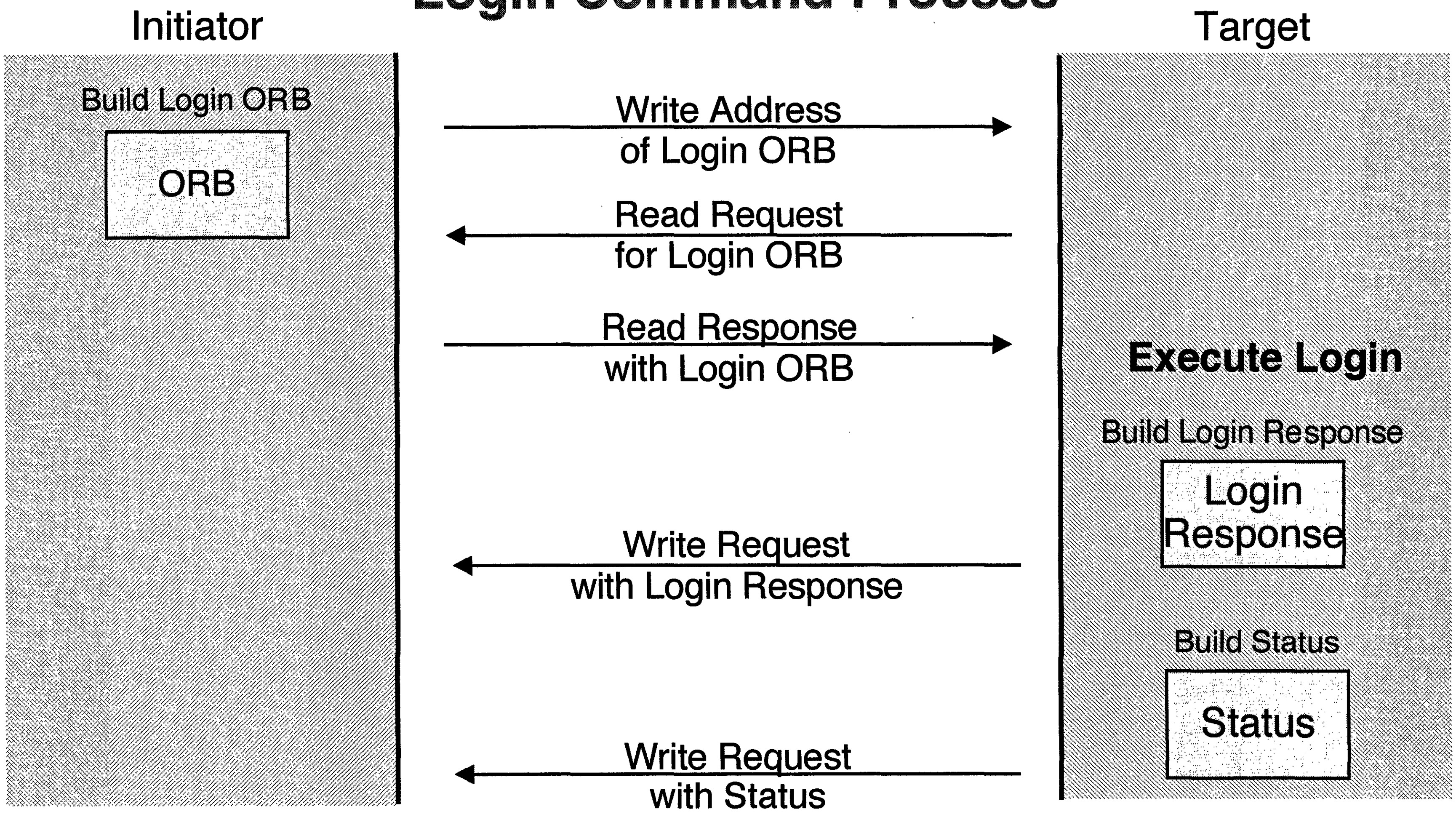
Response informs Initiator location of Command Agent

- **Where to write Command ORB addresses**

Exclusive use provisions

- Only one Initiator logged in at a time

Login Command Process



Initiator

Build Login ORB

ORB

Target

Write Address of Login ORB

Read Request for Login ORB

Read Response with Login ORB

Execute Login

Build Login Response

Login Response

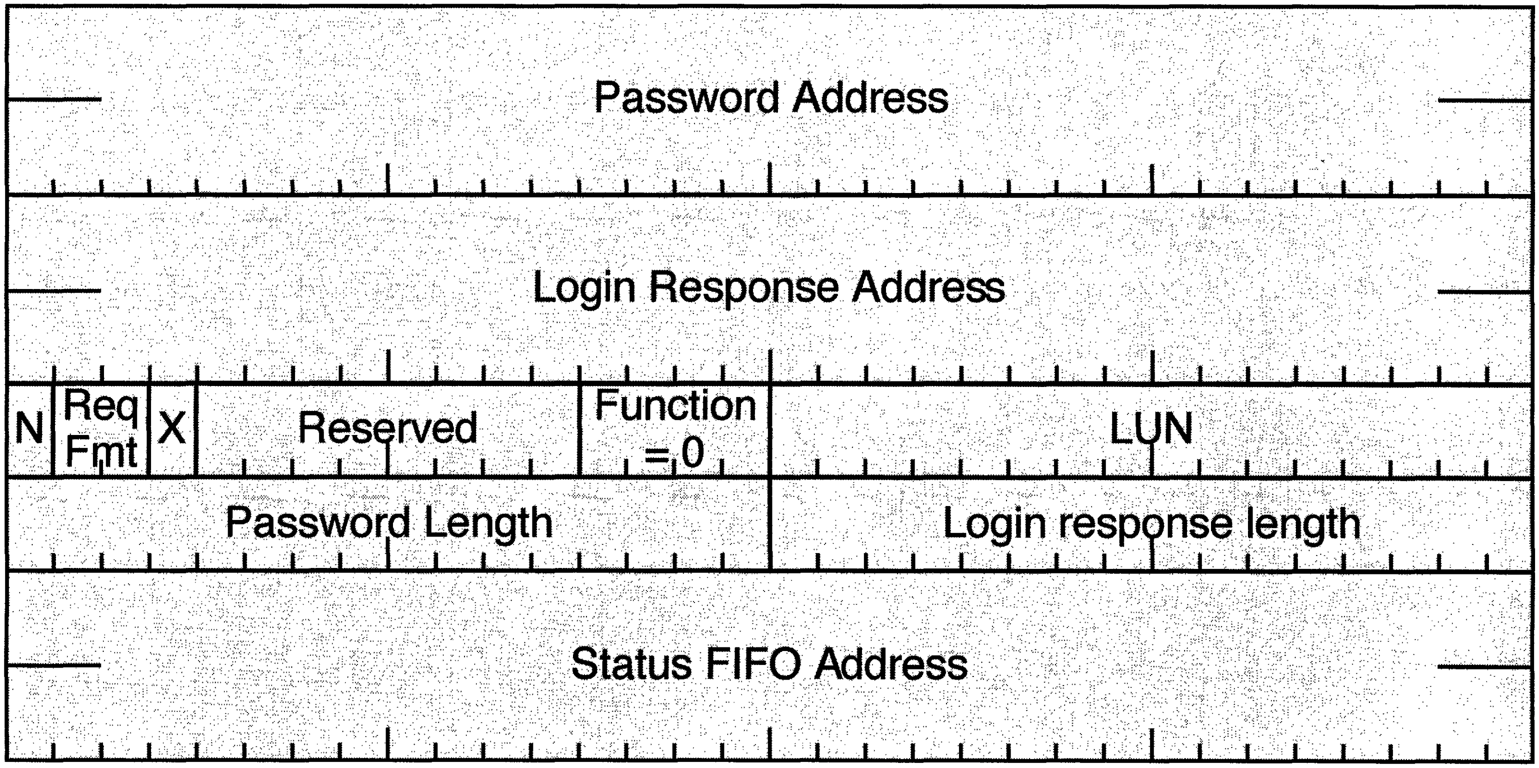
Write Request with Login Response

Build Status

Status

Write Request with Status

Login ORB Format



Password Address					
Login Response Address					
N	Req Fmt	X	Reserved	Function = 0	LUN
Password Length				Login response length	
Status FIFO Address					

Login ORB Fields

Login Response Address	Address in system memory of where to write the Login Response Data
N	Notify Status Flag
Req Fmt	= 00
X	Exclusive Flag 1 = No other Logins to this LUN 0 = Other Logins allowed
Password Length	Length of Password in Bytes If zero, no Password
Password Address	Address in system memory of where to read Password from
Status FIFO Address	Address in system memory of where to write status block

V

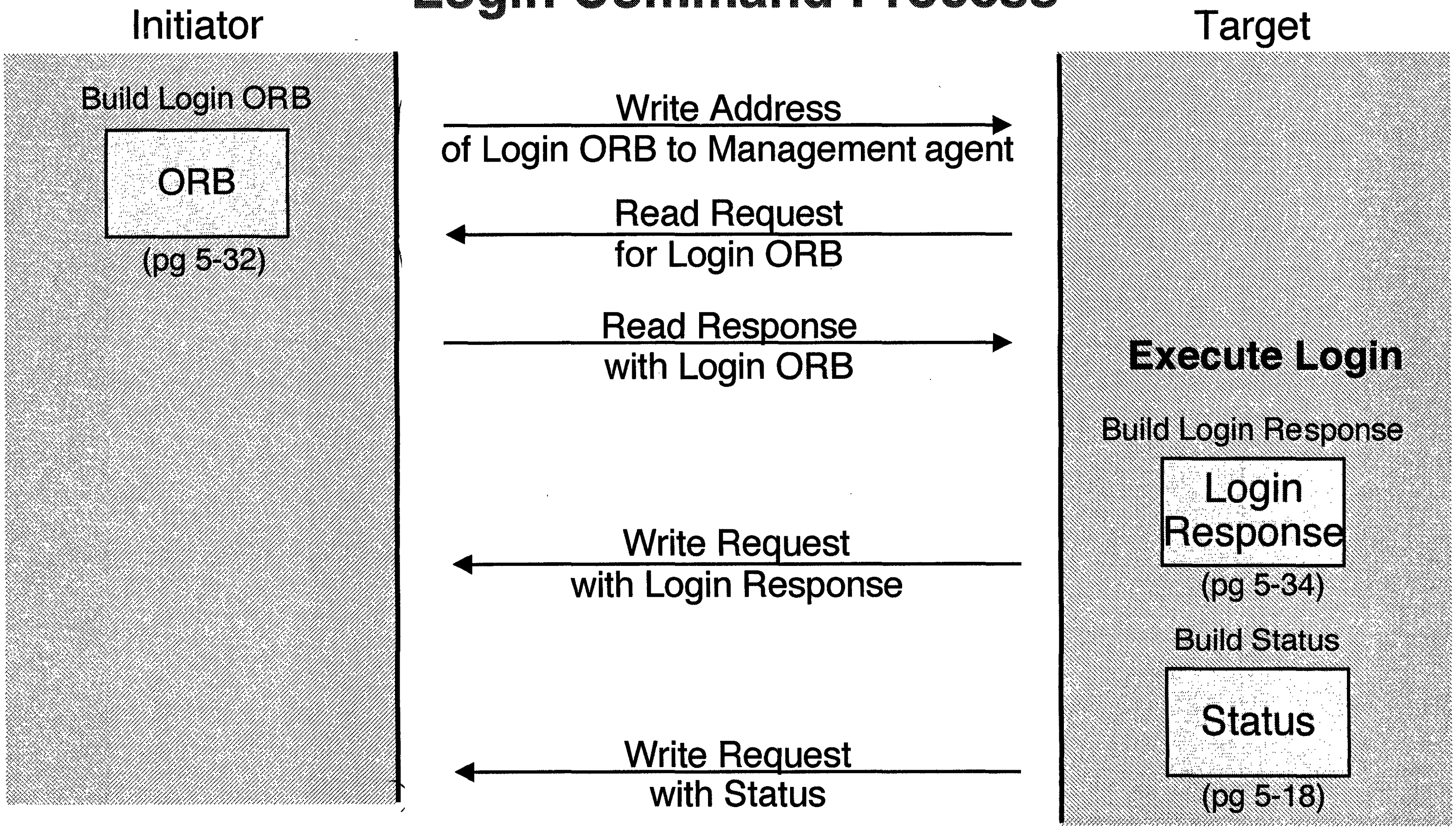
Login

Login Response Packet

Login Response Length = 12 or 16	Login ID
Command Block Agent Address (addr of where Target needs to fetch ORB from)	
Reserved	Reconnect Hold

Login ID	Supplied by Target Used by Initiator in Management ORBs to identify login connection
Reconnect Hold	Specified time target will hold resources waiting for a reconnect following a bus reset Value of 5 means hold resources for 6 seconds

Login Command Process



Initiator

Build Login ORB

ORB

(pg 5-32)

Target

Write Address of Login ORB to Management agent

Read Request for Login ORB

Read Response with Login ORB

Execute Login

Build Login Response

Login Response

(pg 5-34)

Write Request with Login Response

Build Status

Status

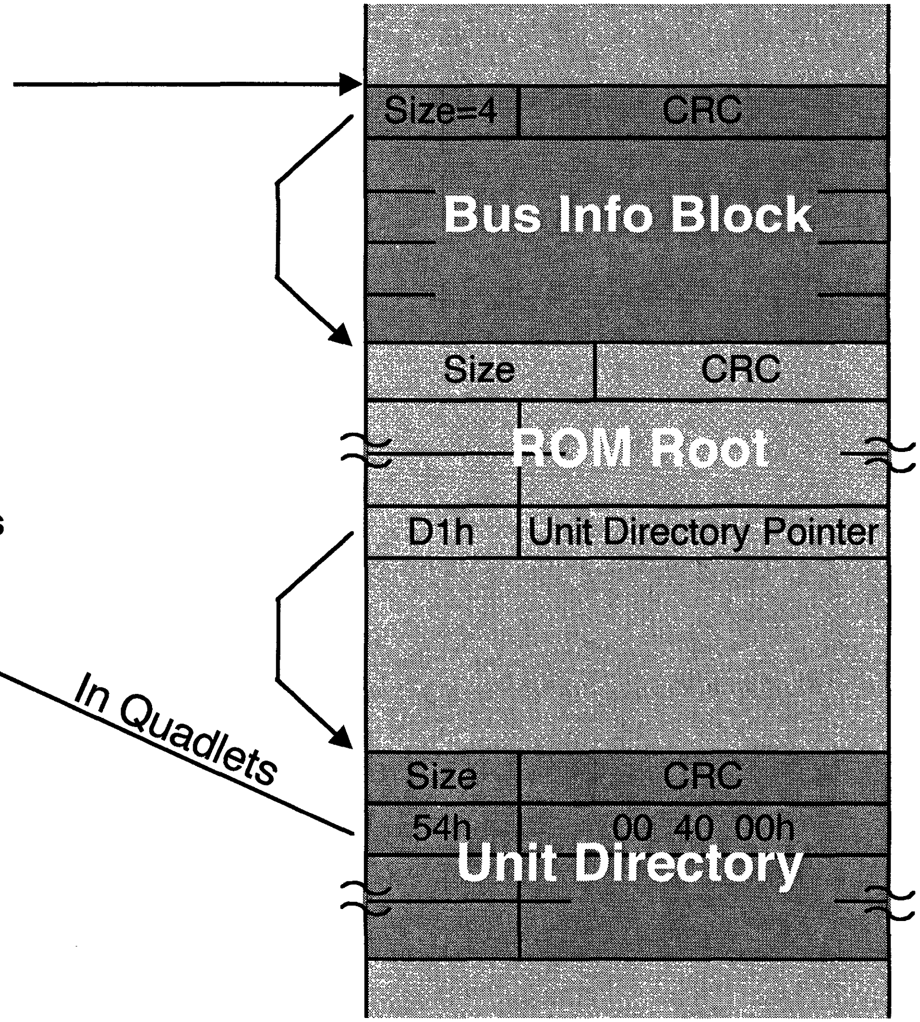
(pg 5-18)

Write Request with Status

How does Initiator know where Management Agent Is?

Find Target's Management Agent Register

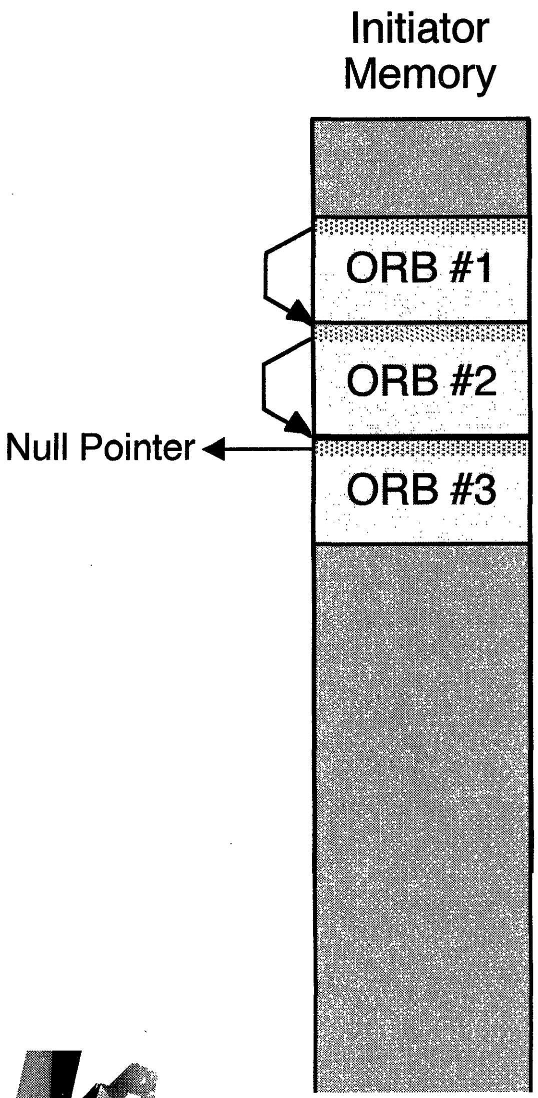
FFFF F000 0400



Management Agent Registers

```
FFFF   F000   0000h
+      0001   0000h
-------------------------
FFFF   F001   0000h
```

Streaming



Initiator creates string of ORBs

Writes Address of 1st ORB to Command Agent

Target executes ORBs

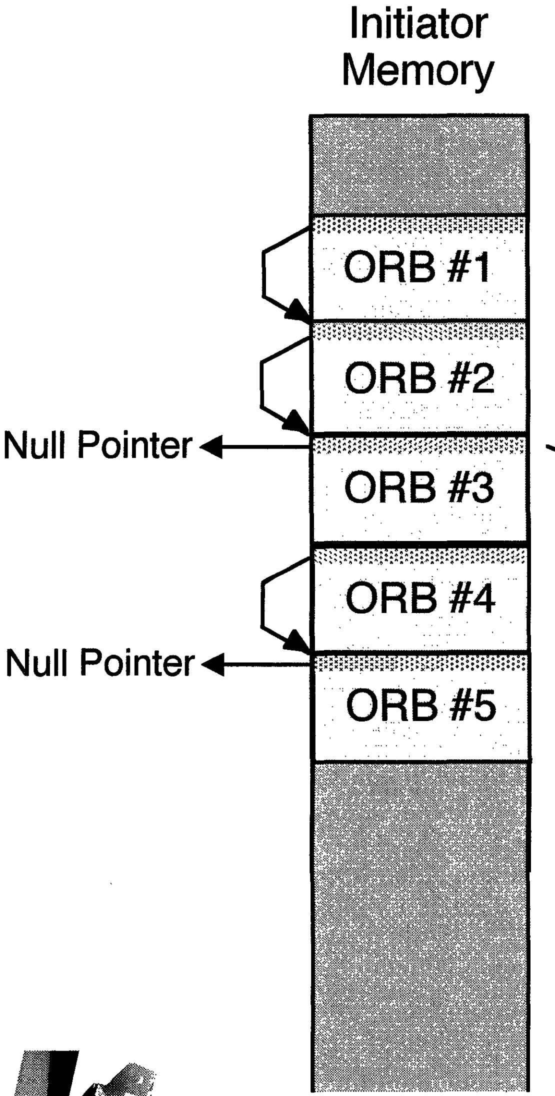
In Order
Out of Order

> **Target Dependent**

Writes Status Block when each ORB Complete

How do you know if the Target executes in order ?

Adding To The Stream



Initiator creates string of ORBs

Writes Address of 1st ORB to Command Agent

Target executing ORBs

Initiator receives two more requests for this Target

Create additional ORBs

Point end of list to next ORB

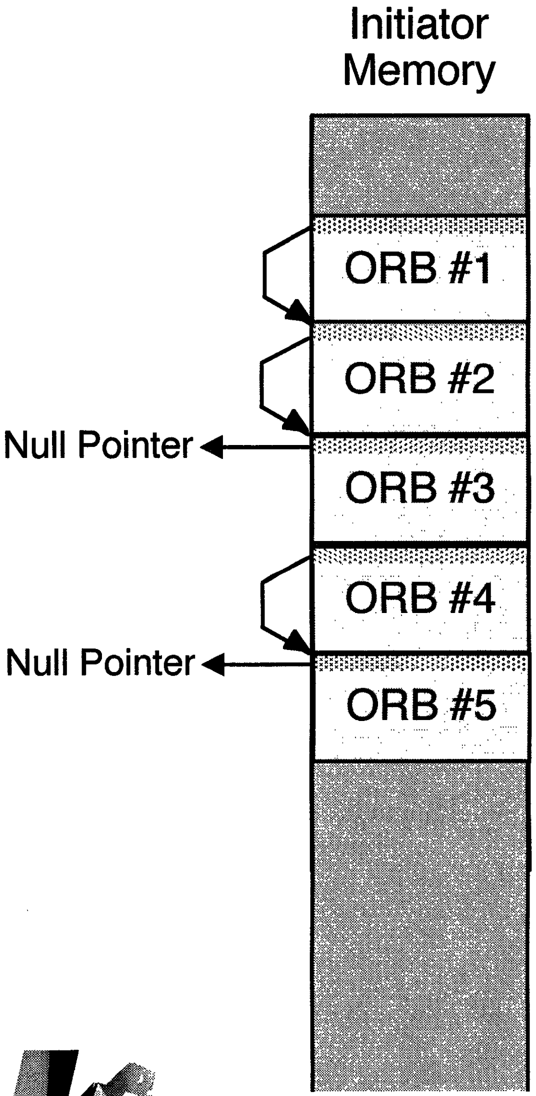
What if the Command Agent has already read it ?

Command Block Agents

Relative offset	Name	Description
00h	Agent State	Reports fetch agent state
04h	Agent Reset	Resets fetch agent
08h	ORB Pointer	Address of request block
10h	Doorbell	Signals fetch agent to refetch an address pointer
14h	Status Acknowledge	Acknowledges receipt of unsolicited status
18h - 1Ch		Reserved for future standardization

Agent States:
0 = Reset
1 = Active
2 = Suspended
3 = Dead

Multiple ORB Streams



Initiator creates string of ORBs

Writes Address of 1st ORB to Command Agent

Target executing ORBs

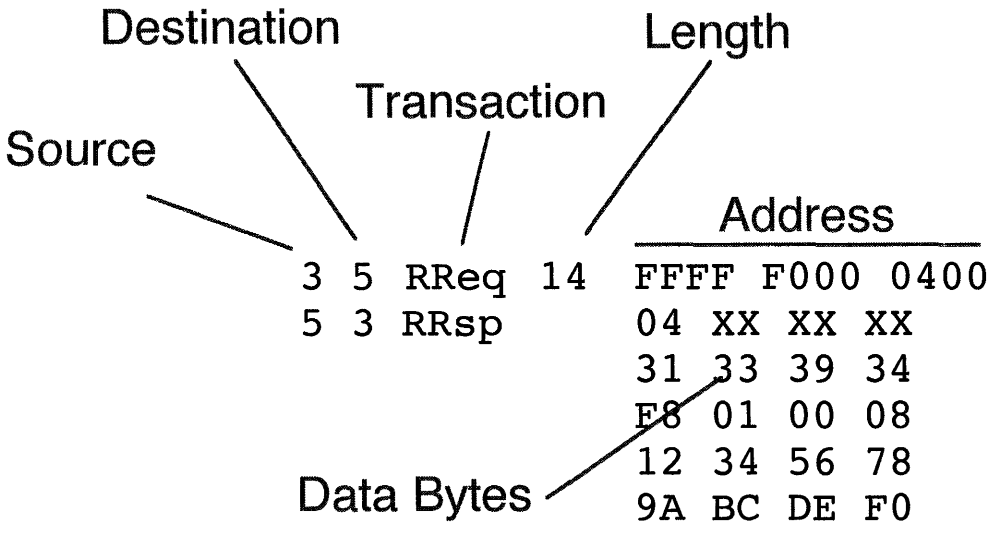
Initiator receives two more requests for this Target

- Create additional ORBs
- Write address to Command Agent

Target can now execute commands from both strings

What if the Command Agent only supports a single ORB pointer?

Tell Us What's Happening - Trace Format



Destination

Length

Transaction

Source

Address

```
3 5 RReq  14  FFFF F000 0400
5 3 RRsp      04 XX XX XX
              31 33 39 34
              F8 01 00 08
              12 34 56 78
              9A BC DE F0
```

Data Bytes

Reason For Request

Key Info Returned

All Numbers In Hex
Trace doesn't show Ack Packets

Tell Us What's Happening - Part 1

beginning of CONFIG ROM

don't Know who's initiator/target yet. anyone can do R-Req. usually targets don't.

```
3 5 RReq 14   FFFF F000 0400
5 3 RRsp      04 XX XX XX
           4  31 33 39 34
           8  E0 FF 80 02
           C  12 34 56 78
          10  9A BC DE F0
```

Read Config ROM

Bus Info Block

```
3 5 RReq  4   FFFF F000 0414
5 3 RRsp      00 04 XX XX
```

Length of root directory

```
3 5 RReq 10   FFFF F000 0418
5 3 RRsp      03 12 34 56
              0C 00 83 80
          20  8D 00 00 02
          24  D1 00 00 04
```

Module vendor id

node capabilities

ptr to node unique id

indirect offset to unit directory

↳ offset from where we are now (FFFF F000 0434)

Root Directory

```
3 5 RReq  4   FFFF F000 0434
5 3 RRsp      00 07 XX XX
```

Length of unit directory

Tell Us What's Happening - Part 2

```
                4x7 quadlets
3 5 RReq  1C  FFFF F000 0438 3-25,3-30                 Read Unit Directory
5 3 RRsp      12 00 60 9E  unit spec ID  ->  SCSI Device 3-35
              13 01 04 83  unit software version
              38 00 60 9E
              39 01 04 D8  -> Reduced Block Cmd p.3-37
              14 0E 00 00  Logical Unit Number
              3A 00 0A 08
              54 00 40 00  management agent CSR = 4000h quad offset
                                                        Unit Directory
                                                        get location of management agent register

3 5 WReq  08  FFFF F001 0000                            Write addr of mangement login ORB to management agent
 local bus    FF C3 00 00  location
 node 3       10 00 00 00  w/in node 3                  addr of login ORB

target in       size of management ORB
control now
5 3 RReq  20  0000 1000 0000 5-26,5-32                  Read Request for login ORB
3 5 RRsp      00 00 00 00 } password
              00 00 00 00 } address
              FF C3 00 00 } login response
              10 10 00 00 } address
     gnrl  -  80 00 00 00
     status
              00 00 00 0C  login response length
              FF C3 00 00 } FIFO Status
              10 20 00 00 - address                     login ORB
```

Tell Us What's Happening - Part 3

```
5 3 WReq  0C  0000 1010 0000
              00 0C 12 34
              FF C5 FF FF
              F0 10 01 00
```

login response addr

login id

node 5

command block agent [illegible]

Login response

address where Target can fetch ORB

```
5 3 WReq  08  0000 1020 0000
              41 00 00 00
              10 00 00 00
```

status FIFO addr

5-18

5-19

addr of ORB the management ORB

Status

command complete Status for this ORB

```
3 5 WReq  08  FFFF F010 0108
              FF C3 00 00
              10 00 00 00
```

initiator in control

cmd agent

ORB ptr & bytes offset from cmd block agent

initiator writes ORB pointer (address)

ORB address

```
5 3 RReq  20  0000 1000 0000
3 5 RRsp      00 00 00 00
              10 00 00 20
              FF C3 00 00
              20 00 00 00
              82 90 00 20
              12 00 00 00
              08 00 00 00
              00 00 00 00
```

target in control

next orb

bfr node ID

bfr addr

data descriptor

control

inquiry cmd

8 bytes

ULP command

SCSI cmd

Command ORBs can be any size, not set @ 20h

target requests to read ORB

ORB

Tell Us What's Happening - Part 4

```
5 3 RReq  20  0000 1000 0020
3 5 RRsp      80 00 00 00
              00 00 00 00
              FF C3 00 00
              20 00 00 20
              82 90 00 20
              00 00 00 00
              00 00 00 00
              00 00 00 00
```

null bit set, last ORB

target fetches next ORB

SCSI-cmd test unit ready cmd

ORB

```
5 3 WReq  08  0000 2000 0000
              0E 00 03 03
              00 00 00 00
```

write inquiry data

~~comp~~ execute cmd in 1st ORB

data

```
5 3 WReq  08  0000 1020 0000
              01 00 00 00
              10 00 00 00
```

status fifo

ORB addr

write status

request complete nothing else to report

What Condition Is The Target In ?

has one outstanding ORB

owes status for test unit ready cmd

Review

1. What does the next ORB pointer point to?
2. What are the limitations on the location of each ORB?
3. What is addressed by the data descriptor field in the ORB?
4. Explain direct addressing
5. Explain indirect addressing
6. Where are the function codes?
7. Explain the login process
8. What is the main information passed in each transaction?

SBP-2 Notes

SBP-2 Notes

Section 6

SCSI Over SBP-2

Subjects Covered

Relationship of SCSI CDB, ORB and 1394 packet

SCSI status block

Messages

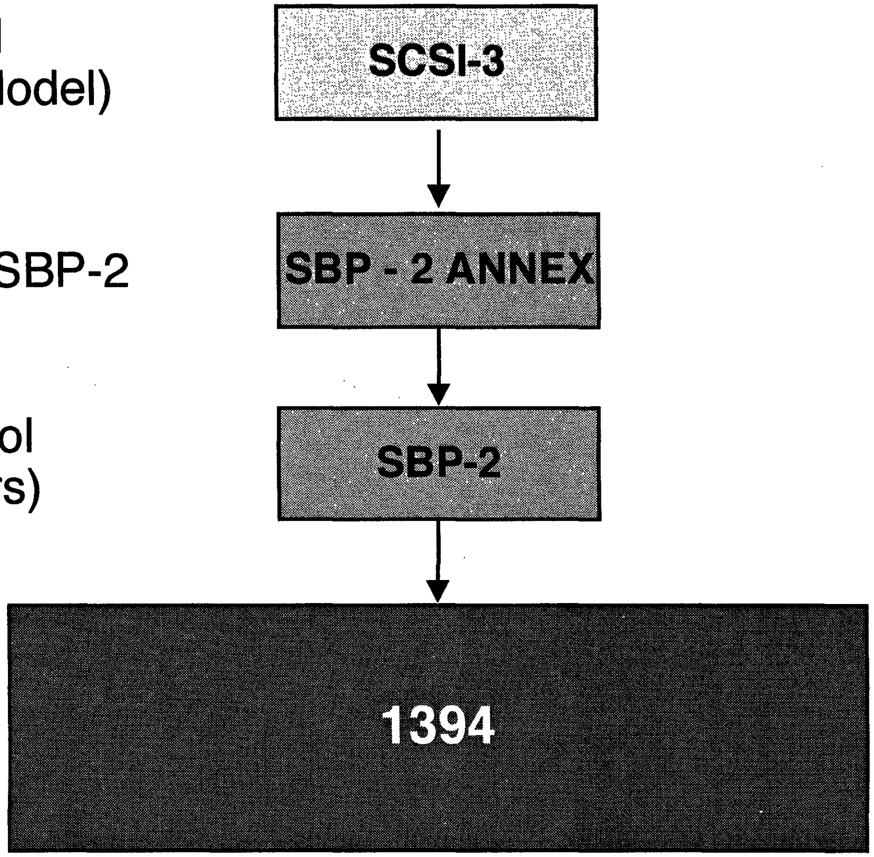
RBC

SCSI on 1394

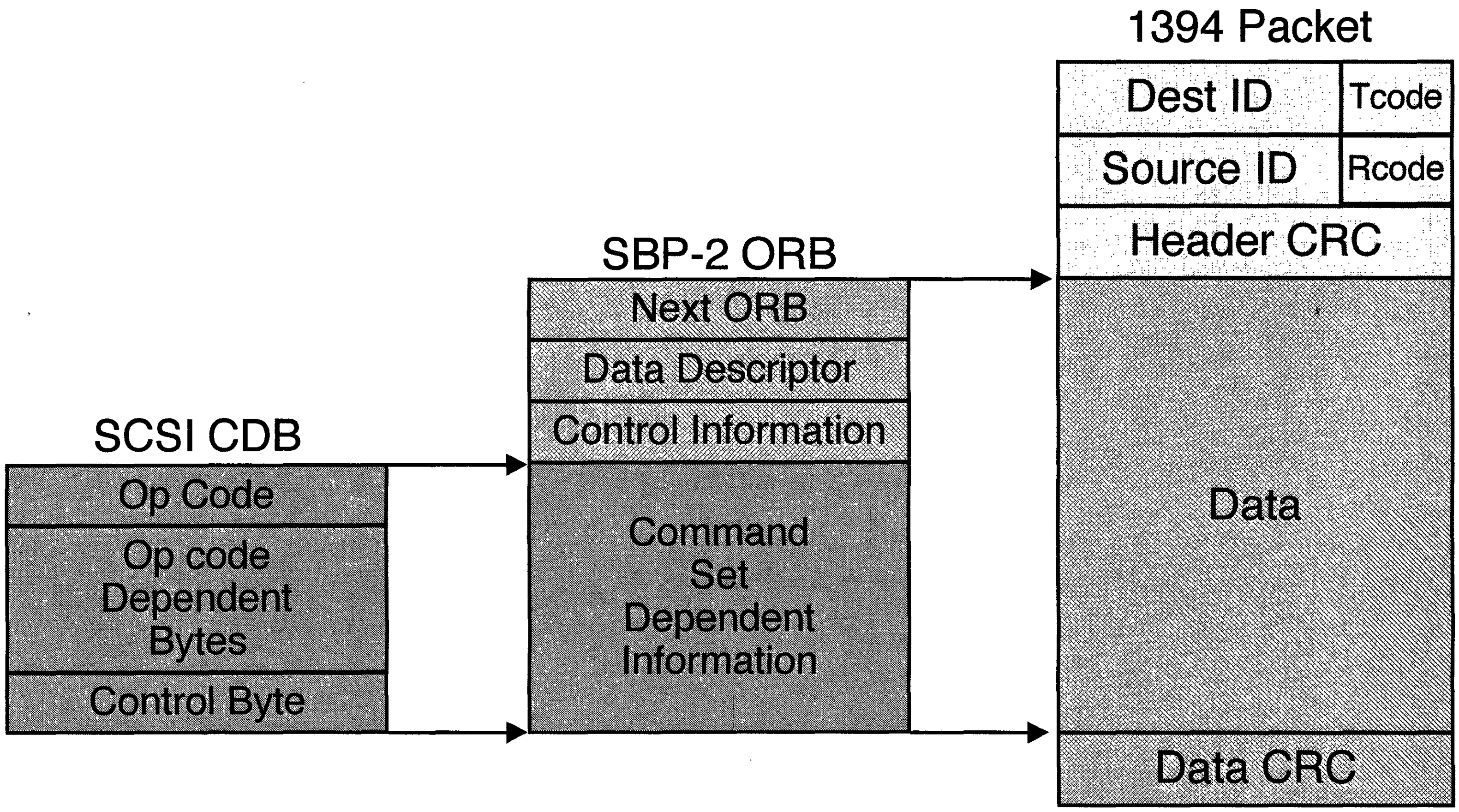
Defined by SAM
(SCSI Architectural Model)

SCSI-3 Transport via SBP-2

Serial Bus Protocol
(The last two hours)



Relationship between 1394, SBP-2 and SCSI



SCSI CDB

Op Code

Op code Dependent Bytes

Control Byte

SBP-2 ORB

Next ORB

Data Descriptor

Control Information

Command Set Dependent Information

1394 Packet

Dest ID

Tcode

Source ID

Rcode

Header CRC

Data

Data CRC

Using SCSI On 1394

Use Config ROM to find Management Agent Address

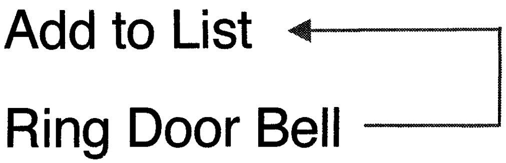
Login In with Management Agent

- Get a Login ID
- Locate Command Agent

Build ORB List

Write ORB List Address to Command Agent

(Watch Status FIFO for completion)



Add to List

Ring Door Bell

SCSI Status

Request Sense Command not needed

- Status returned for each ORB
- No contingent allegiance!

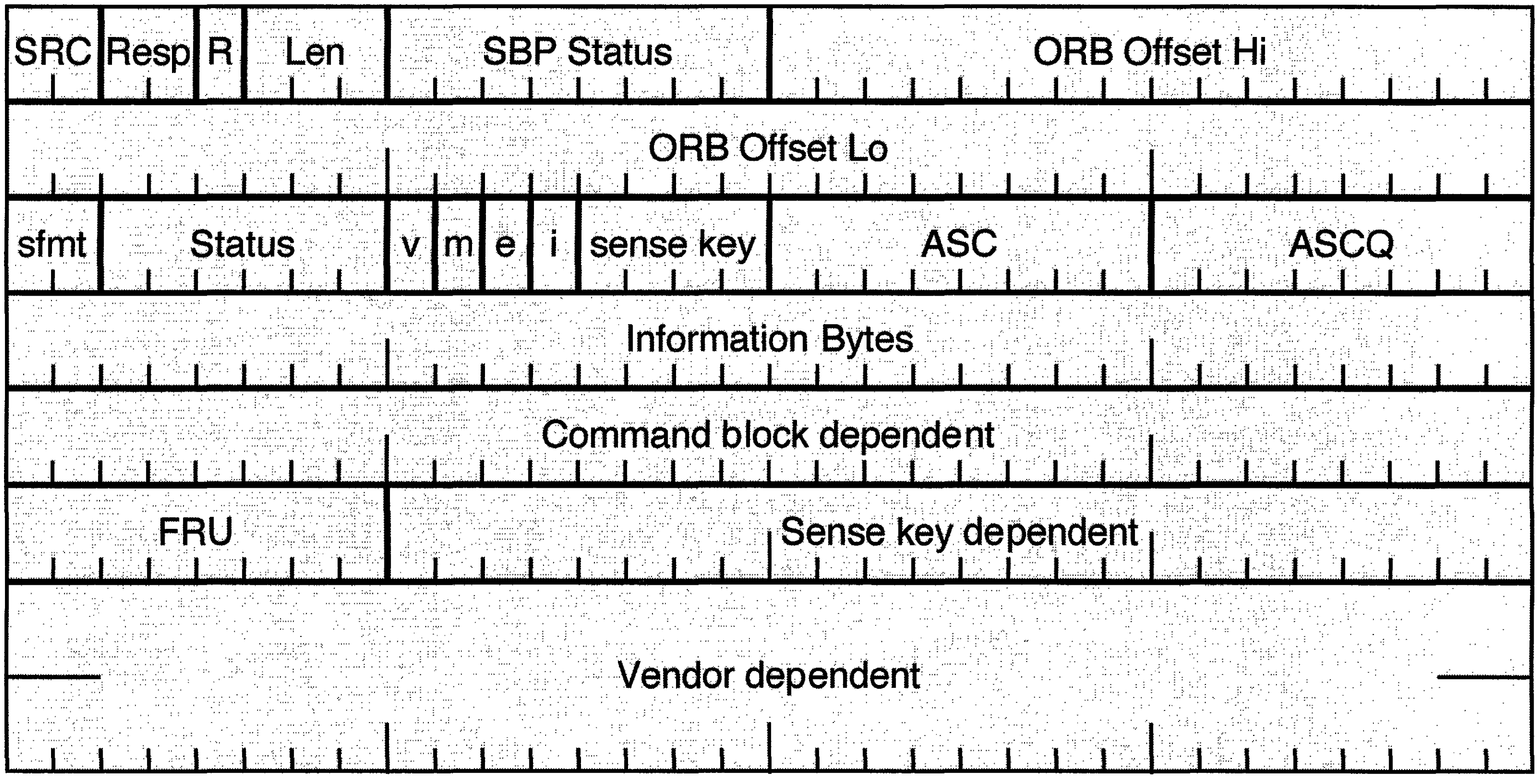
SBP-2 Adopted SCSI Status Format

- Sense Key, ASC, ASC-Q in Status Block

Can use Notify bit to reduce Status Traffic

this page is intentionally blank

Status Block



SRC | Resp | R | Len | SBP Status | ORB Offset Hi

ORB Offset Lo

sfmt | Status | v | m | e | i | sense key | ASC | ASCQ

Information Bytes

Command block dependent

FRU | Sense key dependent

Vendor dependent

Note: If there is no error, the target need only post the first two quadlets of status

Status Block Definitions

SRC

00b = Solicited Status, not end of list
01b = Solicited Status, next ORB = Null
10b = Unsolicited Status
11b = Unsolicited Status, ISOCH Error

Resp Response.

0 = Request complete. The request completed without transport protocol error.
1 = Transport failure. Target detected nonrecoverable transport error.
2 = Illegal request. Unsupported bit or field in ORB
3 = Vendor dependent.

Len Length. Number of valid quadlets -1 stored as status

SBP status

0 = No additional sense to report
1 = Invalid request type
2 = Speed not supported
3 = Page size not supported
4 = Access denied
5 = Logical unit not supported
6 = Maximum payload too small
7 = Too many channels
8 = Resources unavailable
9 = Function rejected
A = Login ID not recognized
FF = unspecified error

V

Status Block Definitions (continued)

sfmt Status format
0 = Current error (SCSI error code 70)
1 = Deferred error (SCSI error code 71)
2 = Reserved
3 = Vendor dependent format

Status This is the command set status (SCSI/ATA/ATAPI)
0 = Good
2 = Check condition
4 = Condition met
8 = Busy
10h = Not supported by SBP-2 devices
14h = Not supported by SBP-2 devices
18h = Reservation conflict
22h = Command terminated
28h = Not supported by SBP-2 devices
30h = Not supported by SBP-2 devices
All other values are reserved for future standardization

v The information stored in the Information quadlet is valid

m, e, l File Mark, end of medium, incorrect length indicator are defined in the applicable command set standards

Status Block Definitions (continued)

Sense Key	Sense key; 0 = No sense 1 = Not ready 2 = Recovered error 3 = Medium error 4 = Hardware error 5 = Illegal request 6 = Unit attention 7 = Data protection 8 = Blank check 9 = Vendor dependent Ah = Not supported by SBP-2 devices Bh = Aborted command Ch = Not supported by SBP-2 devices Dh = Volume overflow Eh = Miscompare Fh = Reserved for future standardization
All other fields	Defined by command set standards

No More Messages

Identify Message	Performed with LUN on Login
	Each LUN has separate Login ID
	(Possible separate Command Agent)
Tagged Queuing	Each ORB tagged with ORB Address
	Device can be Ordered or Unordered
	No mechanism for Ordered Subsequence
Disconnect/Reconnect	Packetized Protocol handles
Address Pointers	Overwrite or Re-Read

SAM Features Not Supported

Asynchronous Event Notification

(SBP-2 does support unsolicited status)

Soft Reset

Untagged Tasks

Linked Commands (or Flag)

NACA BIT

1394 Reduced Block Commands (RBC)

SCSI Device Type = 0E

Subset of 18 SCSI commands for magnetic recording block devices
- Both fixed and removable devices

Based on SCSI Block Commands (SBC) & SCSI Primary Commands (SPC)
- Restricts options and parameters

Initial transport = 1394 with SBP-2 mapping

Proposal to ANSI committee October 1997

Reduced Block Commands

Command	OP Code	Reference
Format Unit	04h	RBC
Inquiry	12h	SPC-2
Mode Select	15h	SPC-2
Mode Sense	1Ah	SPC-2
Persistent Reserve In	5Eh	SPC-2
Persistent Reserve Out	5Fh	SPC-2
Prevent/Allow Media Removal	1Eh	SPC-2
Read (10)	28h	RBC
Read Capacity	25h	RBC
Release	17h	SPC-2
Request Sense	03h	SPC-2
Reserve	16h	SPC-2
Start/Stop Unit	1Bh	RBC
Synchronize Cache	35h	RBC
Test Unit Ready	00h	SPC-2
Verify	2Fh	RBC
Write (10)	2Ah	RBC
Write Buffer	3Bh	SPC-2

Notes: Read (6) and Write (6) are not included
Request Sense optional because of Auto Sense
Details of commands provided in Appendix B

RBC - Event Status Notification

Asynchronous Event Notification
SCSI-2 (AEN)

Asynchronous Event Reporting
SCSI-3 (AER)

Unsolicited Status Sense
RBC

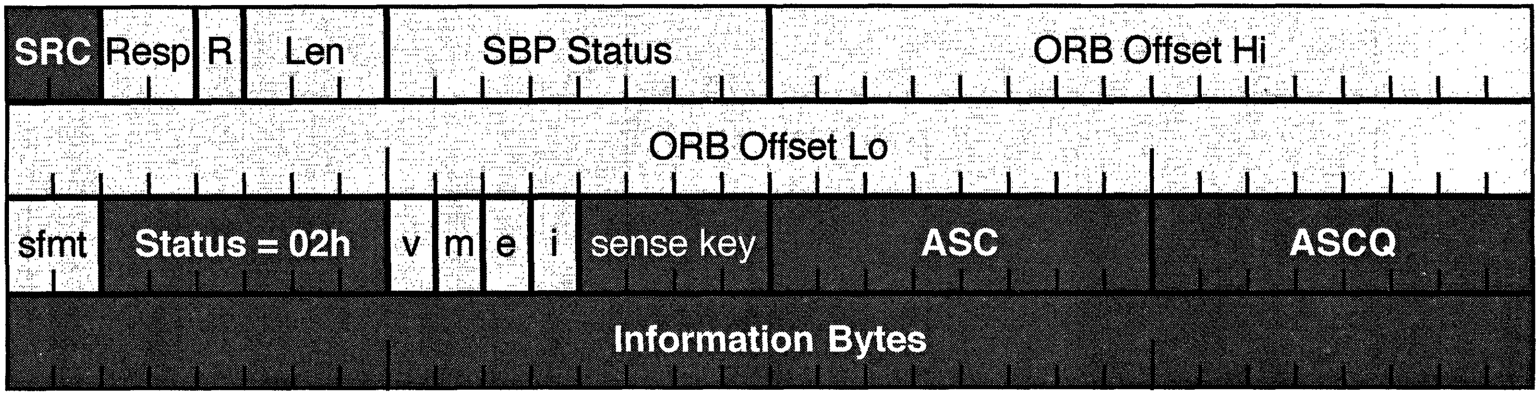
Device returns a Status Block without an ORB request

U (Unsolicited Status) bit in Status Block = 1

This Reports:

Unsolicited Status Sense
Power Management Class Event
Media Class Event
Device Busy Class Event

Unsolicited Status - Determining What Happened



Sense Keys	ASC	Description
2h	04h	Device Not Ready
6h	28h	Not ready to Ready Transition
6h	29h	Power on Reset, bus reset, etc.

6h	7Eh	Notification of an Event
		ASCQ = 02h Power Management Class Event
		ASCQ = 04h Media Class Event
		ASCQ = 06h Device Busy Class Event

Unsolicited Status Sense

RBC
Power Management Information

Byte 0	Byte 1	Byte 2	Byte 3
Event	Status	Reserved	Reserved

Event

00h - No power state change
01h - Device successfully change to the specified power state
02h - Device failed to enter the last requested requested power state
03 - FFh - Reserved

Status

00h - Reserved
01h - Action State
02h - Idle State
03h - Standby State
04 - FFh - Reserved

RBC
Media Event

Byte 0	Byte 1	Byte 2	Byte 3
Event	Status	Start slot	End slot

Event

00h - Media status is unchanged
01h - Eject request
02h - Specified slot has new media
3h - Media has been removed from specified slot - requires user intervention
04 - FFh - Reserved

Status

Bit 1 - Media present
0 - Door or Tray open
2-7 - Reserved

RBC
Device Busy Event

Byte 0	Byte 1	Byte 2	Byte 3
Event	Status	Time (MSB)	Time (LSB)

Event

00h - No event is available
01h - Timeout occurred
02 - FFh - Reserved

Status

00h - No event, Device ready to accept commands
01h - Device waking up
02h - Device completing an earlier command
03h - Device is completing a deferred operation
04 - FFh - Reserved

Review

1. Define how SCSI sense data in mapped into the status block
2. What is the benefit of RBC?
3. How is the SCSI CDB mapped into the 1394 packet?
4. Which t code will be used to move the SCSI CDB?

SCSI Over SBP-2 Notes

Section 7

ATA Over SBP-2

Subjects Covered

IDE/ATA/ATAPI registers

Tailgate

Bridge

Byte ordering

ATA Or IDE

ATA = AT Bus Attachment

Name of the ANSI Standard (X3T13 committee)

IDE = Integrated or Intelligent Drive Electronics

Popular name in the industry

Physical: 40 pin ribbon cable

Supports 2 devices max per cable

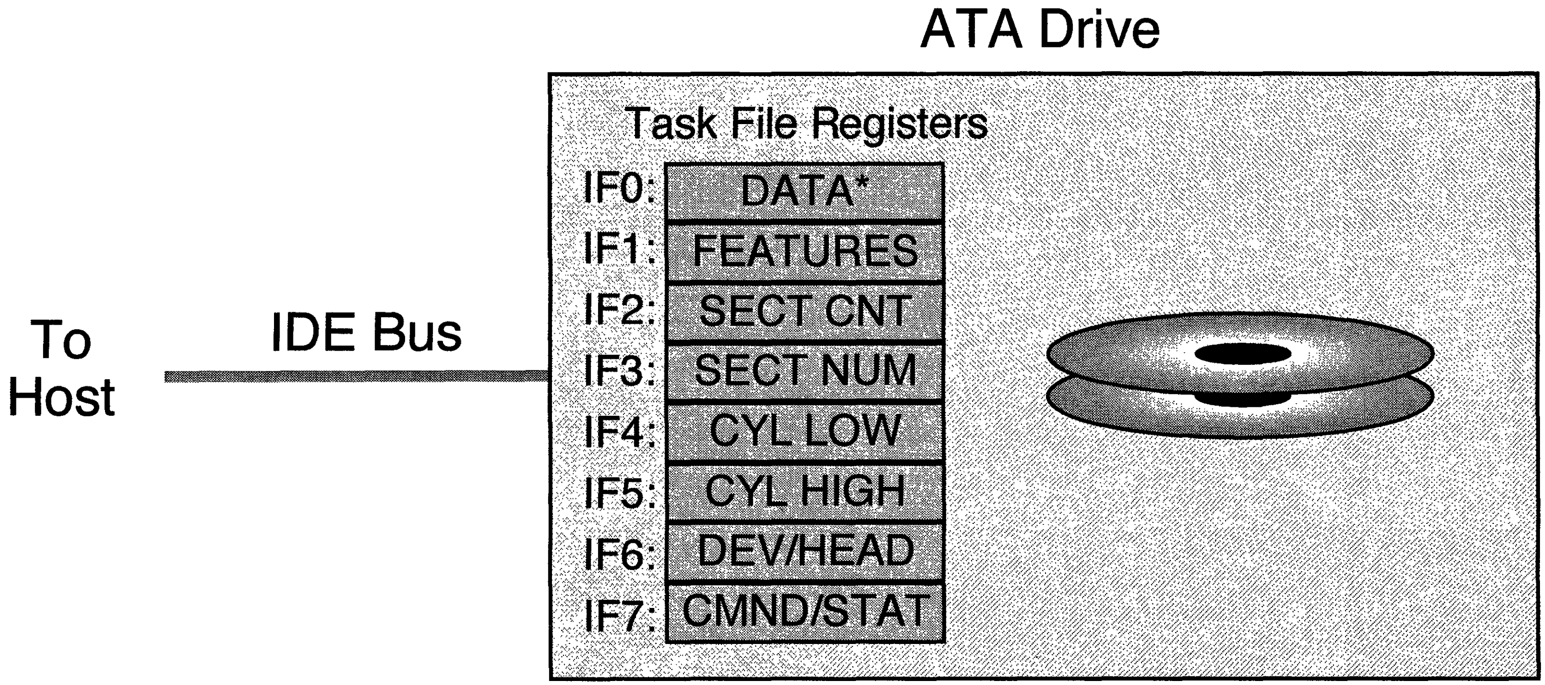
18 inches maximum

Logical: Micro processor has direct access to control registers

PIO = Programmed Input Output

DMA = Direct Memory Access

ATA Task File



ATA Drive

To Host

IDE Bus

Task File Registers

IF0: DATA*

IF1: FEATURES

IF2: SECT CNT

IF3: SECT NUM

IF4: CYL LOW

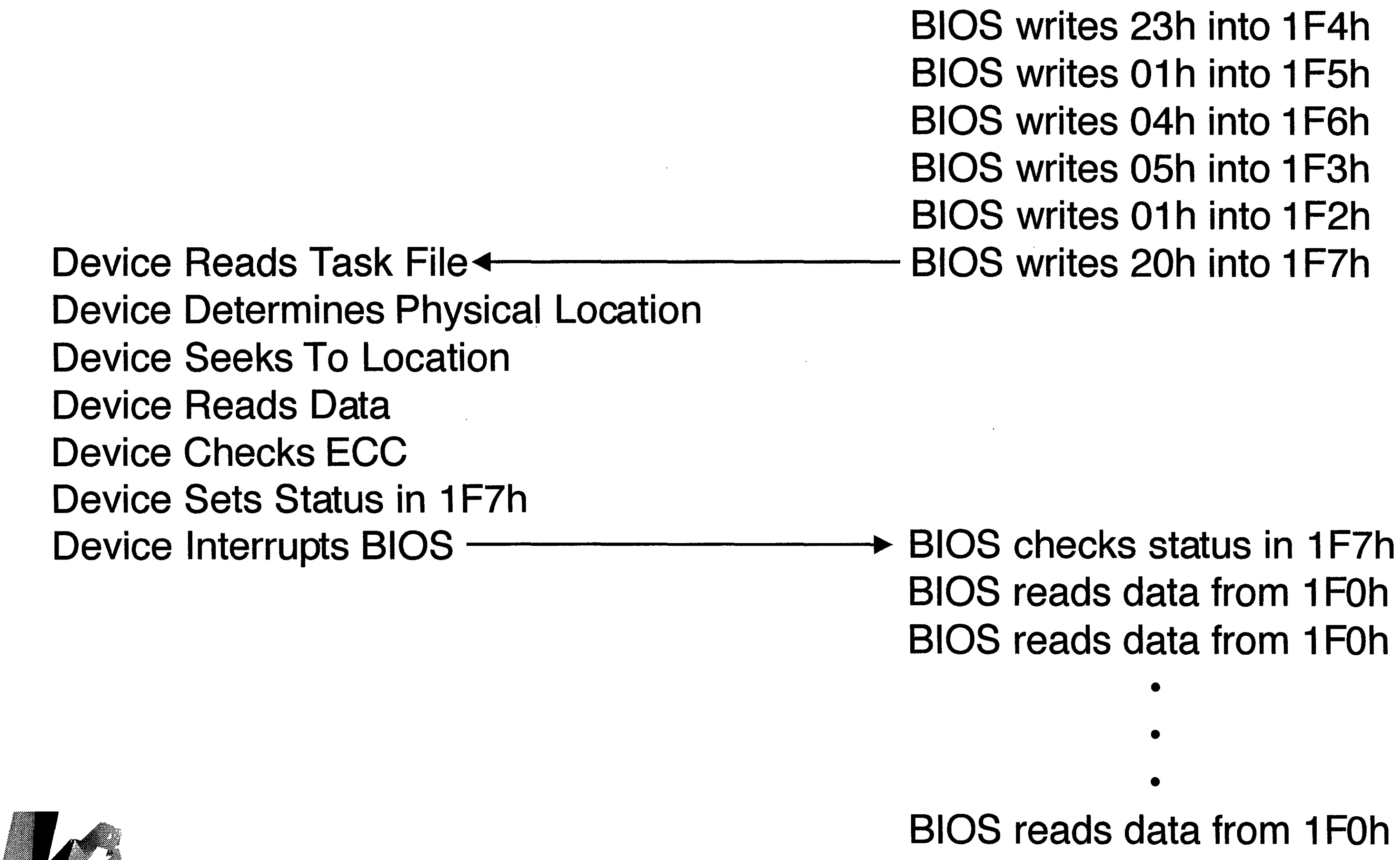
IF5: CYL HIGH

IF6: DEV/HEAD

IF7: CMND/STAT

*Access 16 bits wide

Example ATA Read Command



BIOS writes 23h into 1F4h
BIOS writes 01h into 1F5h
BIOS writes 04h into 1F6h
BIOS writes 05h into 1F3h
BIOS writes 01h into 1F2h
BIOS writes 20h into 1F7h

Device Reads Task File
Device Determines Physical Location
Device Seeks To Location
Device Reads Data
Device Checks ECC
Device Sets Status in 1F7h
Device Interrupts BIOS

BIOS checks status in 1F7h
BIOS reads data from 1F0h
BIOS reads data from 1F0h
•
•
•
BIOS reads data from 1F0h

ATAPI

ATA Packet Interface

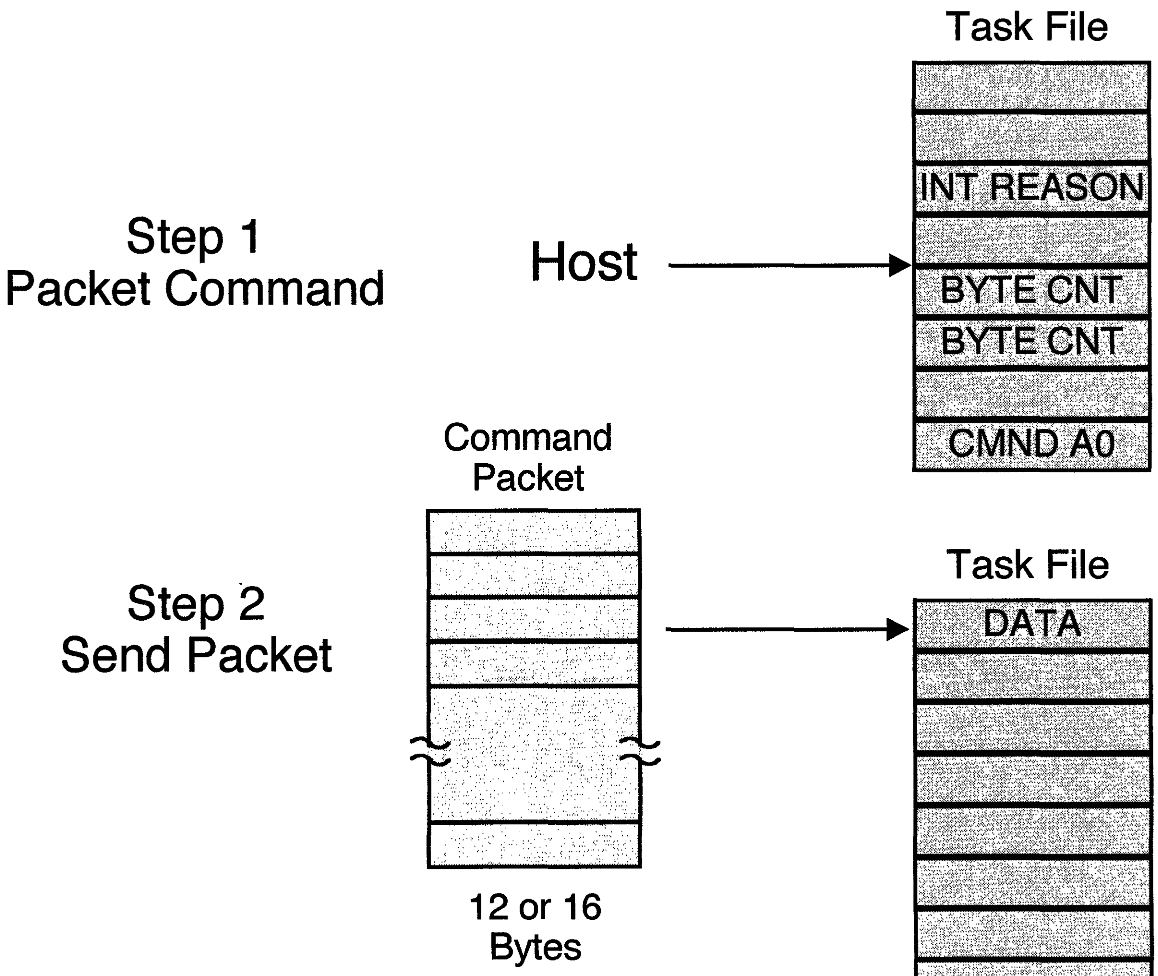
SCSI Command Packets sent over ATA

Popular method for interfacing to CD-ROMs

New ATA Command: Packet Command (A0h)

- “Here, execute this SCSI Command” command
- Packet = 12 or 16 Byte SCSI Command

ATAPI Command Process



Step 1
Packet Command

Host

Task File

INT REASON

BYTE CNT

BYTE CNT

CMND A0

Step 2
Send Packet

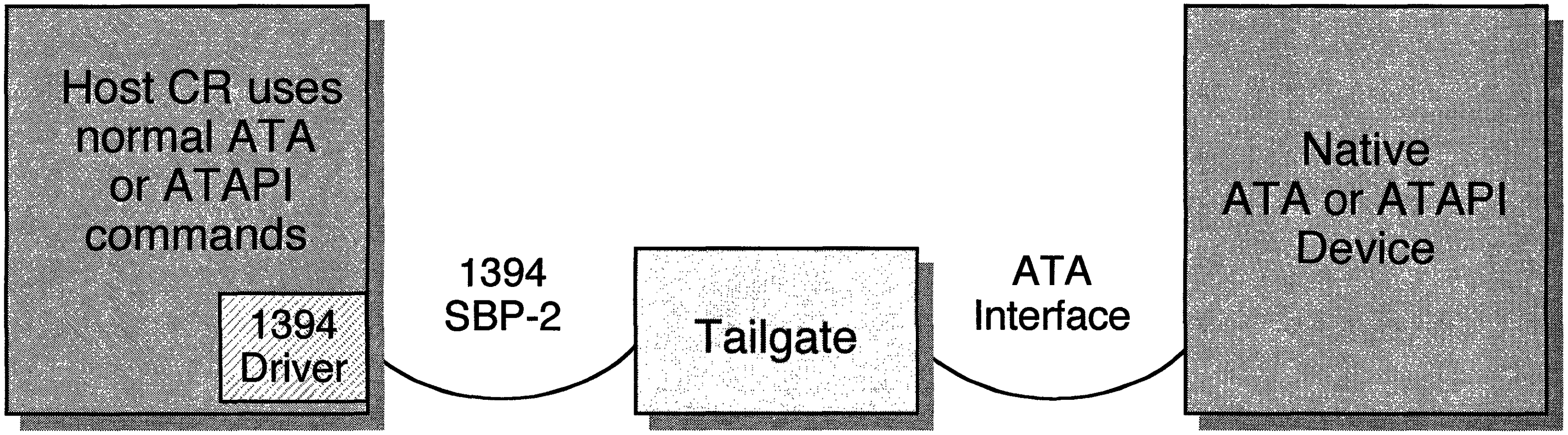
Command
Packet

12 or 16
Bytes

Task File

DATA

Using Native ATA or ATAPI Devices


Host CR uses normal ATA or ATAPI commands
1394 Driver
1394 SBP-2
Tailgate
ATA Interface
Native ATA or ATAPI Device

Translates between 1394 SBP-2 and ATA

Separate board, chip on device, embedded in device controller

Tailgate Characteristics

Low cost

Does not support isochronous

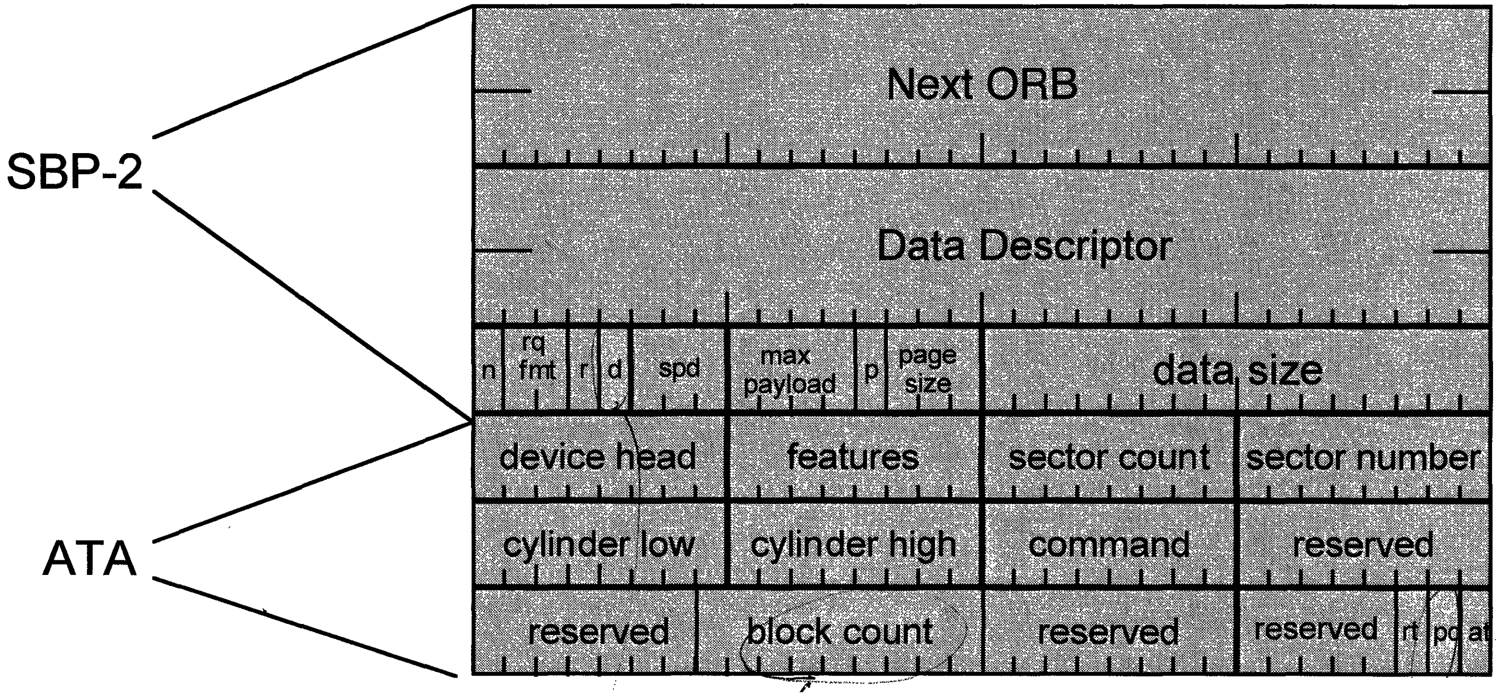
Allows only a single login to each logical unit

Supports either 1 or 2 logical units

PIO block commands (Read/Write Multiple) not supported

Read and Write Long not supported

ATA Command ORB

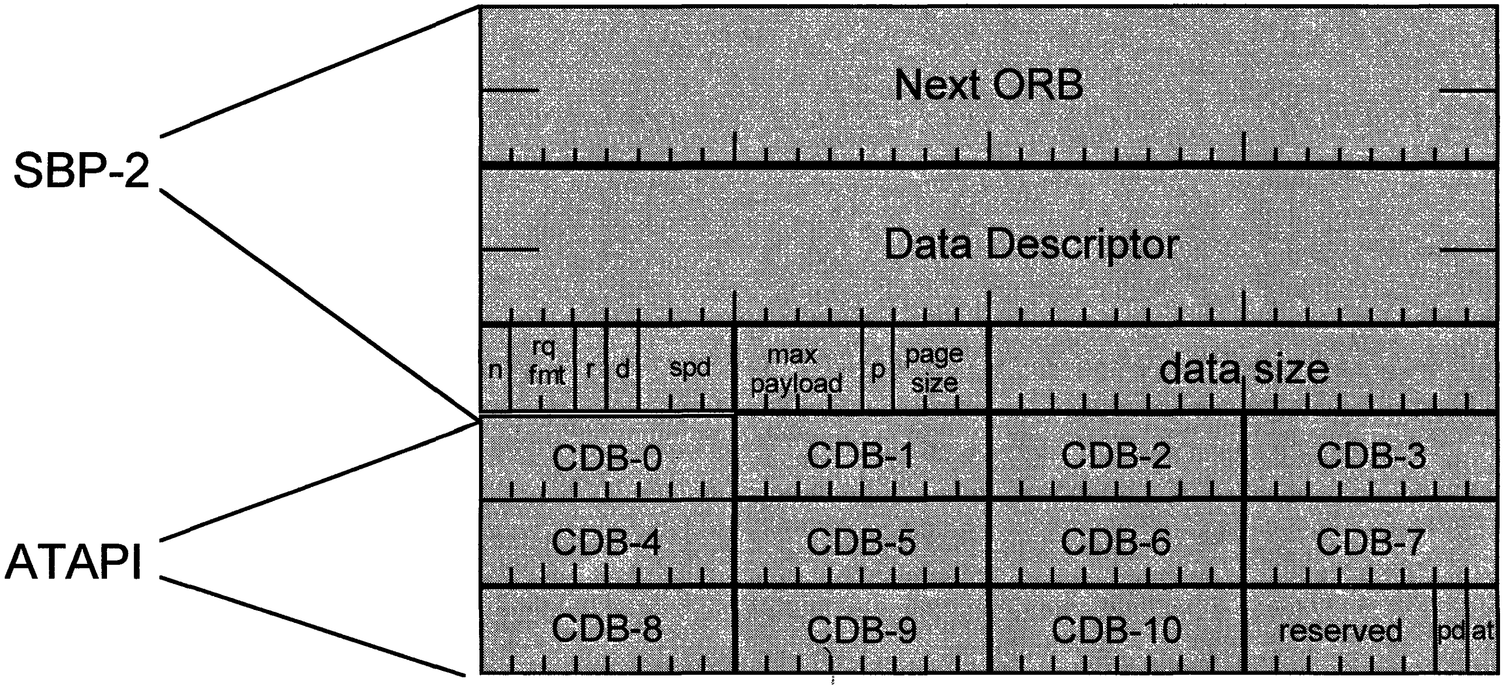


Next ORB			
Data Descriptor			
n rq fmt r d spd	max payload p page size	data size	
device head	features	sector count	sector number
cylinder low	cylinder high	command	reserved
reserved block count		reserved	reserved rt pd at

ATA ORB Definitions

Next ORB	Address of next command ORB in the chain
Data Descriptor	Serial Bus address of data source/destination (quadlet aligned)
n	Notify
rq fmt	0 = SBP-2 command format (1 & 2 not defined)
	3 = Dummy or ABORT command
d	Direction: 0 = use SPB-2 read; 1 = use SBP-2 write
spd	0=S100, 1=S200, 2=S400
max payload	Maximum data length per packet
p	1 = Use page tables
page size	Page table size
data size	Data length, quadlet multiple
rt	0 = Execute command and return status
	1 = Return status only (don’t write task file except to select device)
pd	0 = PIO; 1 = DMA
at	1 = ATA command

ATAPI Command ORB



SCSI CDBs

ATAPI ORB Definitions

Next ORB	Address of next command ORB in the chain
Data Descriptor	Serial Bus address of data serve/destination (quadlet aligned)
n	Notify
rq fmt	0 = SBP-2 command format (1 & 2 not defined)
	3 = Dummy or ABORT command
d	Direction: 0 = use SPB-2 read; 1 = use SBP-2 write
spd	0=S100, 1=S200, 2=S400
max payload	Maximum data length per packet
p	1 = Use page tables
page size	Page table size
data size	Data length, quadlet multiple
rt	0 = Execute command and return status
	1 = Return status only (don't write task file except to select device)
pd	0 = PIO; 1 = DMA
at	0 = ATAPI command

Tailgate Status Block

<table>
<tr><td rowspan="2">SBP-2
Info</td><td>SRC</td><td>resp</td><td>r</td><td>len</td><td>SBP status</td><td colspan="2">ORB offset hi</td></tr>
<tr><td colspan="7">ORB offset lo</td></tr>
<tr><td rowspan="2">Tailgate
Info</td><td colspan="4">Device/head</td><td>Error</td><td>Sector cnt</td><td>Sector num</td></tr>
<tr><td colspan="4">Cylinder lo</td><td>Cylinder hi</td><td>Status</td><td>Tailgate Status</td></tr>
</table>

If a failure occurred before command completed

SBP-2 Info contains the relevant information

If the failure was at the Tailgate or Device level

Tailgate Info contains the relevant information

Status Block Definitions

SRC	00b	Solicited Status, not end of list
	01b	Solicited Status, next ORB = Null
	10b	Unsolicited Status
	11b	Reserved

ORB Offset Identifies ORB for this status

Resp Response

0 = Request complete. The request completed without transport protocol error.

1 = Transport failure. Target detected nonrecoverable transport error.

2 = Illegal request. Unsupported bit or field in ORB.

3 = Vendor dependent.

r Reserved (set to 0)

V Len Length. Number of valid quadlets -1 stored as status

SBP Status

Indicates status from the transport level:

0	=	No additional sense to report
1	=	Invalid request type
2	=	Speed not supported
3	=	Page size not supported
4	=	Access denied
5	=	Logical unit not supported
6	=	Maximum payload too small
7	=	Too many channels
8	=	Resources unavailable
9	=	Function rejected
A	=	Login ID not recognized
FF	=	unspecified error

If anything other than 0, Tailgate Info will be 0

Tailgate Status

Value	Description
0h	No error
1h	Data size not exact (informative)
2h	No ATAPI command phase
3h	Busy at start of command
4h	Task aborted
5h	Task set aborted
6h	Tailgate reset has completed
7h - FEh	Reserved
FFh	Other protocol errors

This page left blank

ATA Map 02

New mapping to replace Tailgate

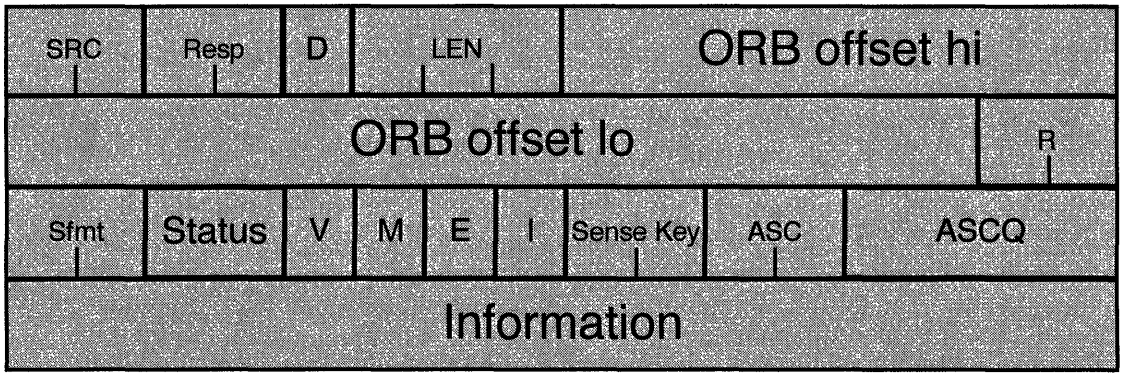
Uses SCSI Host Driver

Bridge Device translates SCSI commands to ATA commands for ATA devices, and passes SCSI commands for ATAPI devices.

Command Mapping

SCSI Command	ATA Command	Op Code
Mode Select (10)	Idle	E3
Mode Sense (10)	Identify Device	N/A
Read (10)	Read DMA or	C8 or
	Read Sectors	20
Start/Stop Unit	Seek & Standby	70 &
	Immediate	E0
Synchronize Cache	Flush Cache	E7
Test Unit Ready	None	
Write & Verify (10)	Write Verify	3C
Write Buffer	Download Microcode	92
Write (10)	Write DMA or	CA or
	Write Sectors	30

Status



Sfmt	=	0
Status	=	00 - only 1st 2 quadlets sent
		02 - check condition
	=	08 - busy
V	=	0
M & E	=	0
I	=	ATAPI = 0
	=	ATA = attempted to move more than 256 blocks
Sense key	=	ATAPI from request sense
ASC	=	ATAPI from request sense
ASCQ	=	ATAPI from request sense
Information	=	0

Status Sense Key And Code Meanings

Sense Key	ASC	ASCQ	Error
2	04	00	Device not powered
6	29	00	Unit attention, reset
5	25	00	LUN not 0
5	24	00	LBA or Transfer Length out of range or Rel Addr or Byte clk bits set
2	04	01	DRDY bit set before issuing cmd
2	04	01	BSY bit set before issuing cmd
B	00	00	Transport failure during cmd execution
4	00	00	ERR & ICRC set at the completion of cmd
3	11	00	ERR & UNC set at the completion of cmd
3	21	00	ERR & IDNF set at the completion of cmd
3	12	00	ERR & AMNF set at the completion of cmd
3	00	00	DF was set at the completion of the cmd
5	20	00	ABRT was set at the completion of the command

Review

1. Define what a tailgate is in reference to 1394
2. Contrast the tailgate and the bridge

ATA Over SBP-2 Notes