X3T9.2/91-5

P1394 High Speed Serial Bus
A Technical Summary

This document is a techical summary of the proposed IEEE P1394 High Speed Serial Bus based on draft 3.2 of the standard and other working group documents.

written by Michael D. Teener, P1394 Chair

Sunday, January 13, 1991

1. Introduction

The P1394 standard describes a high speed serial bus designed for low cost yet providing the data transfer speed and low latency needed for a peripheral bus or as a backup to a traditional parallel backplane bus. The highlights of the Serial Bus include:

1) A physical layer supporting both cable media and many ANSI/IEEE standard 32-bit busses. This includes the proposed ANSI/IEEE standards P1196-1990 NuBus Revison, P896.1 Futurebus+ and P1596 Scalable Coherent Interface.
2) Variable speed data transmission with a standard speed of almost 40 Mbit/sec over cable lengths greater than 10 meters. A higher speed 160 Mbit/sec version is also under development, although it has not been characterized at this time.
3) Both fair and priority arbitration mechanisms with all nodes guaranteed at least partial access to the bus, regardless of priority.
4) Bus transactions that include both block and single quadlet reads and writes, as well as an "isochronous" mode which provides a low-overhead guaranteed bandwidth service.
5) Dynamic address assignment that does not require switches or a physical "slot number."
6) Consistent with the IEEE P1212 Control and Status Register Architecture Specification

The draft P1394 standard is maintained by the chair of the working group, Michael Teener (408-974-3521). Copies of the related P1212 standard can be obtained from Kinkos Copy Service at xxxx, although this may change at any time since P1212 has almost completed it's standardization process.

1.1. Serial Bus Applications

There are three primary applications that have driven the architecture and design of the Serial Bus: an alternate for a parallel backplane bus, a low-cost peripheral bus, and as a bus bridge between architecturally compatible 32-bit busses.

1.1.1. Alternate Bus

There are five primary reasons for providing a serial bus on a system that already has a parallel bus:

1) The system is built out of modules that are based on different backplane bus standards, yet it must all operate as a cohesive whole.

2) The system is too large or physically dispersed to have only a single backplane, yet modules in the different backplanes must communicate. This is called "inter-crate" communication.

3) One or more communicating modules of a system are not located on the backpanel. This is called "off-crate" communication.

4) The system requires a certain level of failure tolerance. In particular, a redundant communication path can diagnose and isolate errors without using the failed parallel bus.

5) Many of the modules in a system are particularly price-sensitive and do not need the full bandwidth of a parallel bus.

1.1.2. Low Cost Peripheral Bus

The Serial Bus can also be used as a powerful and low cost peripheral interconnect. The cable is a simple shielded twisted pair with a small connector and yet the bandwidths are comparable with existing I/O interconnect standards. The Serial Bus has the added advantage of architectural compatibility with 32-bit parallel computer busses. This can impose less overhead than limited-function dedicated I/O interconnects.

1.1.3. Bus Bridge

As mentioned above, the Serial Bus can be used in multiple-bus system configurations. The Serial Bus architecture limits the number of nodes on any bus to 63, but supports bus bridges which connect two or more buses together. The addressing structure follows the P1212 CSR standard and so has sufficient address space to support 1023 buses.

In normal operation, a bus bridge eavesdrops on the bus transactions, but ignores all of the transactions to local addresses. For transactions to remote addresses, a bus bridge is the agent which forwards the packet to the adjacent bus. After initialization, the bus bridges are transparent to the normal system operations.

Although the Serial Bus may be used in many bus configurations, it is expected to be used mostly in hierarchical bus topologies, as illustrated in Figure 1-1 below, where bus #5 is a Serial Bus and bridges together busses 1 through 4.

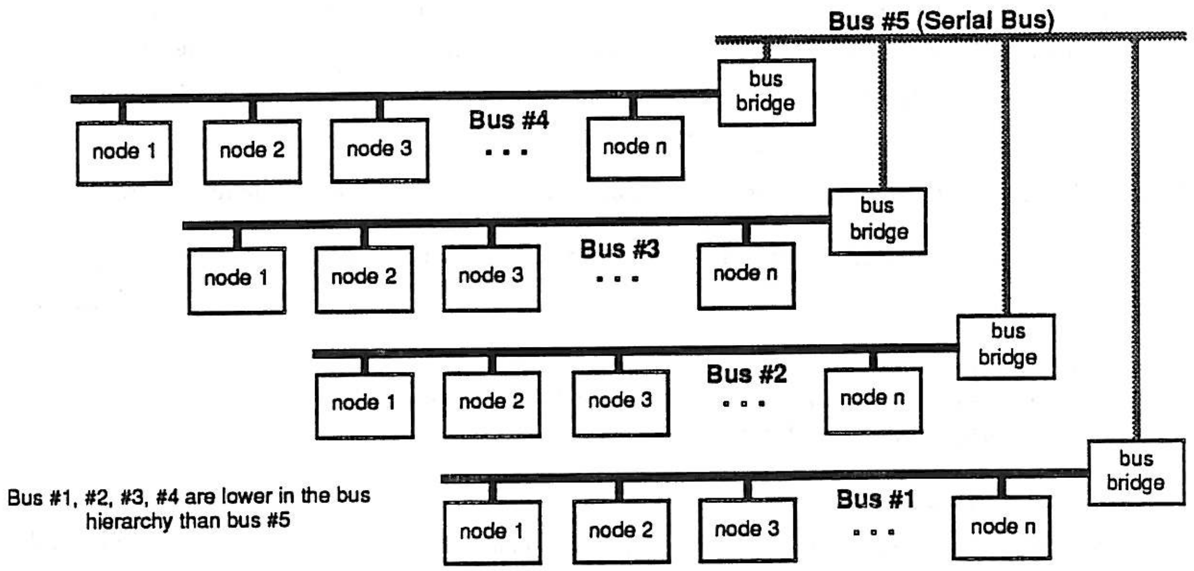


Figure 1-1. Example hierarchical bus topology

1.2. Document Notation

The terms half-word, word, and double-word are avoided in the Serial Bus and CSR descriptions. These definitions are dependent on the processor's word size, which could easily be 16, 32, or 64 bits. Instead, the Serial Bus notation uses the conventions established by previous IEEE bus standards, which are independent of the processor's word size. This notational convention is illustrated below:

Size	32-bit Word Notation	IEEE Std. Notation
1	Byte	Byte
2	Half-word	Doublet
4	Word	Quadlet
8	Double	Octlet

The Serial Bus uses big-ending ordering for byte addresses within a quadlet, and quadlet addresses within an octlet. For 32-bit quadlet registers, byte 0 is always the most significant byte of the register. For a 64-bit quadlet-register pair, the first quadlet is always the most significant. The field on the left (most significant) is transmitted first, and within a field the most significant bit is also transmitted first. This ordering convention is illustrated below:

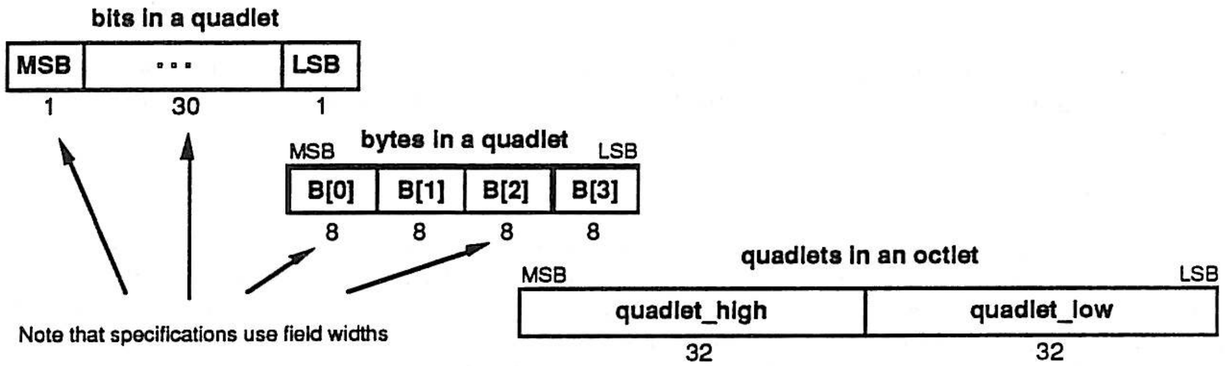


Figure 1-2. Bit and byte ordering

Although the P1394 standard is defined to be big-endian, their data values may also be processed by little-endian processors. To minimize the confusion between conflicting notations, the location and size of bit fields are specified in terms of their widths, rather than their absolute positions, as is also illustrated above.

1.3. Topology

The physical topology of the Serial Bus is divided into two parts as shown in figure 3-1. The first part is called the "backplane environment" and there are several different versions corresponding to the different backplane bus standards. The other part is called the "cable environment" and is completely specified in section 6. Nodes on a single bus may reside in different backplane environments, or directly in the cable environment. There is no requirement that the Serial Bus have any particular set of environments. All nodes may reside strictly in a single backplane, or they all may be directly attached to the cable.

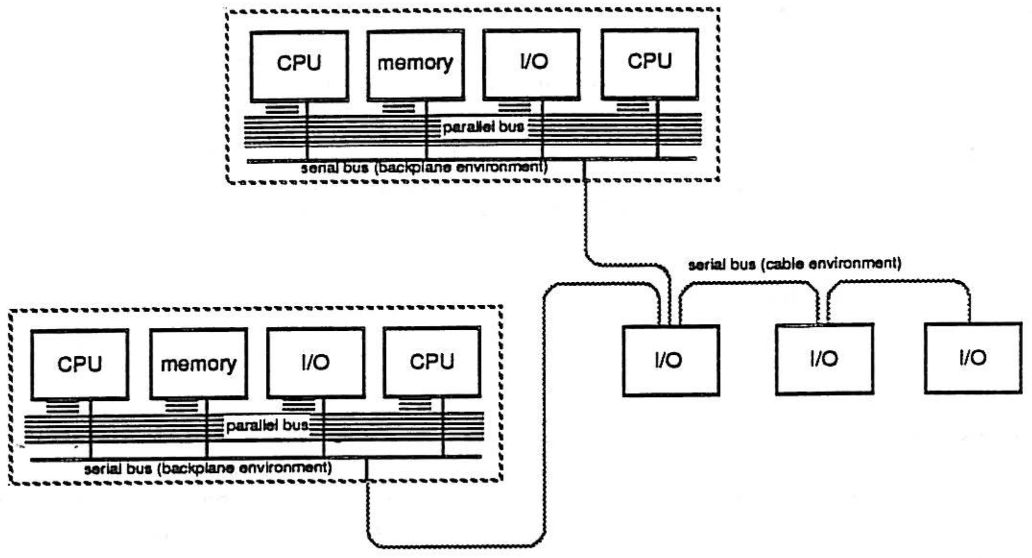

CPU
memory
I/O
CPU
parallel bus
serial bus (backplane environment)
serial bus (cable environment)
I/O
I/O
I/O

Figure 1-3. Serial Bus Physical Topology

Note that this physical topology does not describe anything beyond a single bus. The actual implementation of multiple bus systems using bus bridges is not defined: only the addressing and transactions are specified.

1.3.1. Cable Environment

The physical topology for the cable environment is a non-cyclic network with limited branches and extent. The media consists of 3-pair shielded cables with terminators, transceivers and simple logic dedicated to each port. The trasceivers attach together at the nodes to provide a very short "silicon bus". The cable and transceivers act as bus repeaters between the silicon busses to simulate a single logical "dominant mode" bus. This dominant mode is needed since the bitwise arbitration method used by the Serial Bus requires that each node receive the "OR" of the transmissions of all nodes.

Since it will frequently be inconvenient for low power serial devices to have a separate power wiring system, the media will also have a power pair carrying *30 VDC at no more than 1.5 A*[1]. The actual current available is system-dependent.

1.3.2. Backplane Environment

The Serial Bus can be extended within each physical device as a two-signal electrical bus. This can provide a simple and direct way for internal processing resources to communicate with appropriate peripherals.

The backplane interface to the Serial Bus will make use of the two pins reserved for a Serial Bus by the various ANSI/IEEE bus standards. These two pins are redefined either as a transmit-receive pair or a dominate mode differential pair. Drivers and receivers for these signals follow the conventions established by the appropriate parallel bus standard: e.g., Futurebus using BTL, Fastbus and SCI using ECL, and NuBus using an adaption of the Serial Bus cable environment. Even in systems without standard busses, the electrical equivalent can be routed wherever needed to provide a low cost interconnect.

[1]Parameters that are still being developed by the working group are shown in italics.

1.4. Node and Board Architectures

The Serial Bus architecture is defined in terms of entities called nodes. A node is an addressable entity, which can be independently reset and identified. More than one node may be co-located on a single module, and more than one function may be co-located on a single node. This architectural concept is illustrated below:

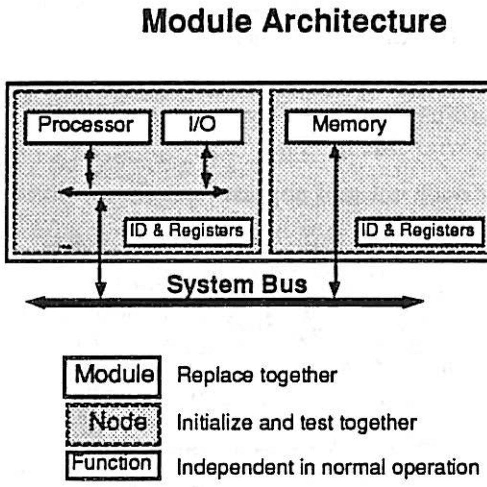


Figure 1-4. Module architecture.

1.5. Addressing

The Serial Bus follows the P1212 standard for 32-bit addressing, where the upper sixteenth of the address space is reserved for node-specific addressing. This allows a system of up to 1023 busses each with 63 nodes to be accessable with a direct and well-known address. This standardization is continued within the node, with 4096 bytes divided between core P1212 resources, registers specific to the Serial Bus, a ROM ID area, and node-specific resources.

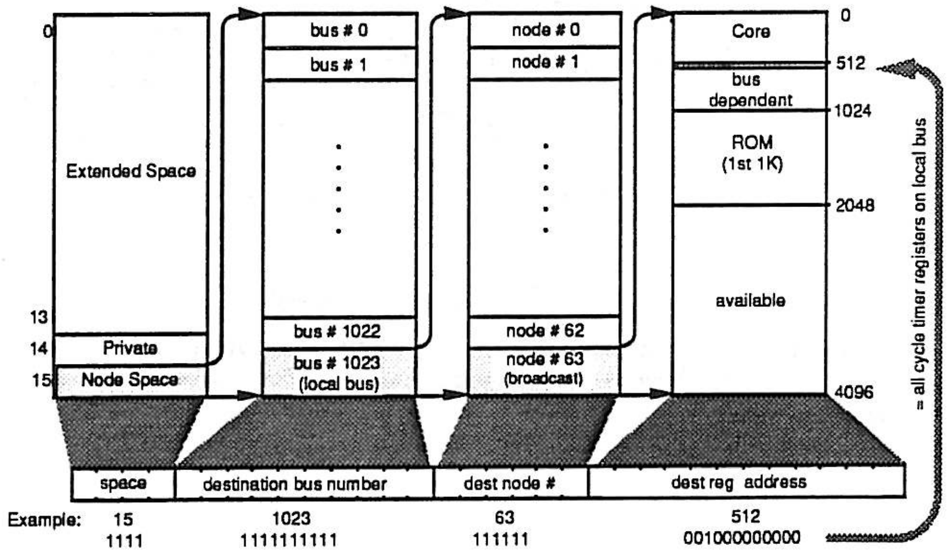


Figure 1-5. Serial Bus Addressing

115

1.6. Protocol Architecture

The serial bus protocols are described as a set of three stacked layers as shown in figure 3-4:

1) The Transaction Layer defines a complete request-response protocol to perform the bus transactions required to support the P1212 architecture.

2) The Link Layer defines a one-way data transfer service to the Transaction Layer. It provides access to the medium, addressing, data checking, and data framing. One Link Layer transfer is called a "subaction".

3) The Physical Layer translates the logical symbols used by the Link Layer into actual physical signals on the different Serial Bus media.

Note that the Transaction Layer does not add any services for isochronous data.

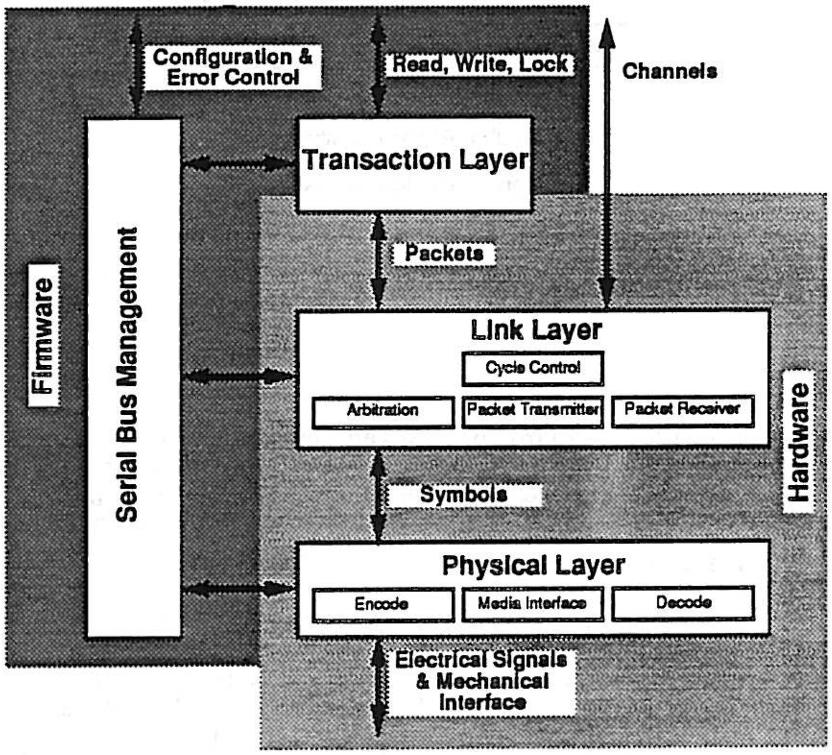


Figure 1-6. Serial Bus protocol stack

2. Transaction Layer

Data is transferred between nodes (a requester and one or more responders) on the serial bus by three different types of transactions, where each transaction consists of one or two subactions: a request and possibly a response. The three transactions are:

1) Get — data is transferred from a responder back to a requester[1].

[1]This could also be called a "read", but it is useful to emphasize that the Serial Bus uses split transactions, so the "get" term was chosen. Note that there can only be one responder. Broadcasts have no meaning in Serial Bus Get transactions.

2) Set — data is transferred from a requester to one or more responders[1].
3) Lock — data is transferred from a requester to a responder, operated on by the responder, and then transferred back to the requestor.

Transactions actually have four actions:

1) Request — the action taken by a requestor to start the transactions.
2) Indication — the reception of a request by a responder.
3) Response — the action taken by the responder to finish the transaction.
4) Confirmation — the reception of the response by the requestor.

These actions and their relation to the data flow is shown below:

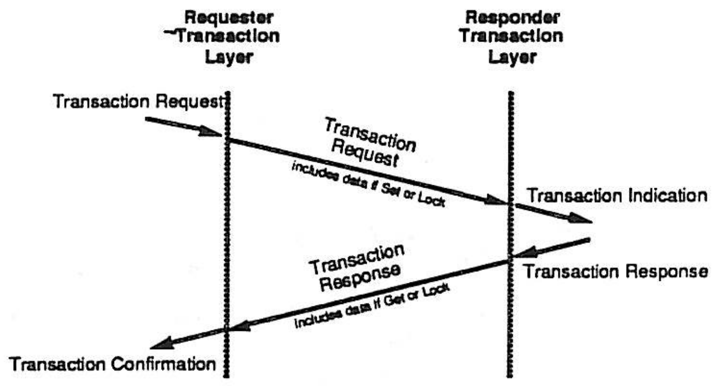


Figure 2-1. Transaction operations.

Transactions can exchange arbitrary amounts of data up to 2048 bytes with arbitrary alignment (no address restrictions). Implementations, however, are allowed to place restrictions on this. In particular, if isochronous link-layer services are implemented, the maximum packet length is restricted to 62 µsec. This means that packets are limited to 256 bytes at 49 Mbaud and 1024 bytes at 196 Mbaud. The only required transactions are quadlet set/get on quadlet aligned addresses (these are the transactions necessary to access the standard CSR resources).

2.1. Lock Subcommands

Since the Serial Bus supports split transactions, it cannot be easily locked while transaction sequences implement indivisible test_and_set operations. Therefore, special lock transactions are defined, which communicate the intent from the requester to the responder, thus allowing the indivisible updates to be performed at the responder. There is one standard lock transaction format, but several different subcommands define conditional and unconditional update actions.

The lock subcommands are based on the implementation model necessary to implement the fetch_and_add and compare_and_swap primitives. The other subcommands define other update actions which can be easily performed with minimal additions to the basic lock-operation hardware. In the lock implementation model two data values (**data** and **test**) are sent in the lock request; one data value (**old**) is returned in the lock response. These are illustrated in Figure 2-2.

[1]Similarly, this could be called a "write". In broadcast Serial Bus Set transactions, the responders do not actually generate a response.

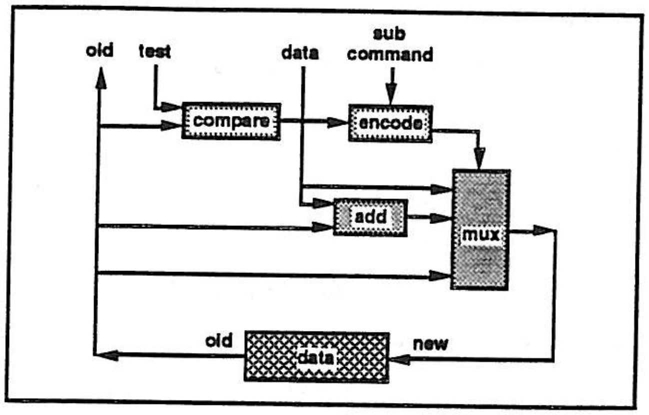


Figure 2-2. Simplified lock model

The three data values (data, test, and old) are all quadlets.

3. Link Layer

The Link Layer provides access to the bus and a one-way data transfer with simple acknowledgement. This transfer is called a "subaction", and there are two types used in the Serial Bus Link Layer:

1) Asynchronous Subaction — 0 to 2048 bytes of data and an upper layer control block are transferred to an explicit address. The subaction has three parts:
   - Arbitration Sequence — a node that wishes to become a source performs a bit-serial arbitration process to gain control of the bus. This is implicit for isochronous transfers.
   - Packet Transmission — a bus transfer code, addresses of the source and destination nodes, and data are sent by the source node. Isochronous packets only include the transfer code and data, the addresses are implicit.
   - Acknowledgement — a uniquely addressed destination will return a "complete", "pending", or one of two "rejected" acknowledgements. Asynchronous broadcast packets do not have acknowledgements.

2) Isochronous Subaction or "Channel" — 0 to 256 bytes are transferred without explicit addressing. Channel transfers use a form of time division multiplexing to provide an implicit address. Note that there is no arbitration or acknowledgement for channels.

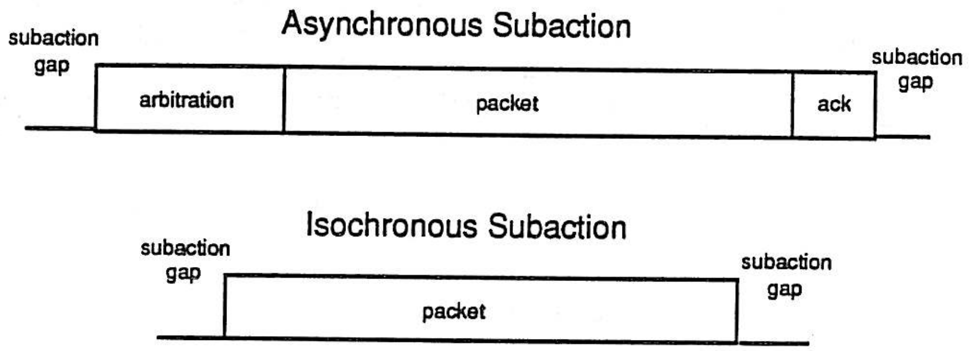


Figure 3-1. Subactions.

The link layer operations also have the request, indication, response, and confirmation actions:

1) Request — the action taken by a link requestor to transmit a packet to a link responder.
2) Indication — the reception of a packet by a link responder.
3) Response — the transmission of an acknowledgement by a link responder.
4) Confirmation — the reception of the acknowledgement by the link requester.

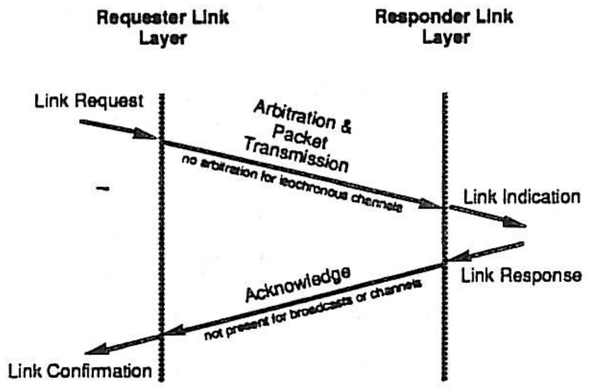

Requester Link Layer
Responder Link Layer
Link Request
Arbitration & Packet Transmission
no arbitration for isochronous channels
Link Indication
Link Response
Acknowledge
not present for broadcasts or channels
Link Confirmation

Figure 3-2. Link Layer operations.

The Transaction Layer and Link Layer interact in a way that optimizes the use of the bus. In particular, transactions can be implemented in two different ways: unified or split. The simplest transaction, a Set with no errors, can be implemented with a single Link Layer subaction: the transmission of a packet and the corresponding acknowledgement as shown below.

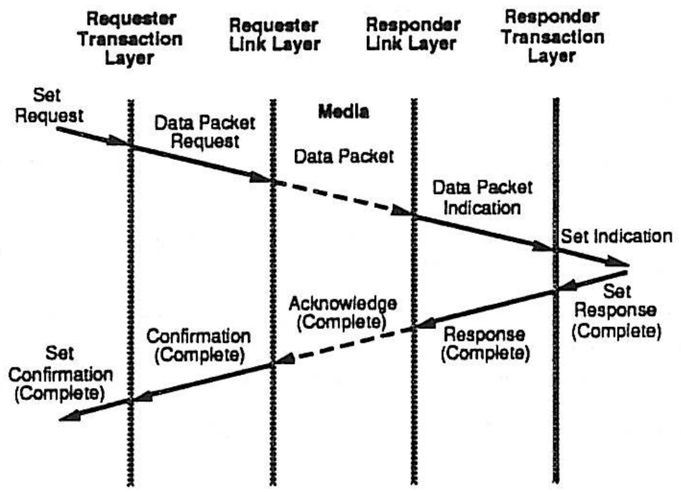


Figure 3-3. Unified Transaction Example.

For Get and Lock transactions and Set transactions that have error status, or where the response takes longer to generate, then a split transaction is required with separate Link Layer subactions for the request and the response. Note that other Link Layer subactions occur on the bus between the request and response subactions of a single transaction.

119

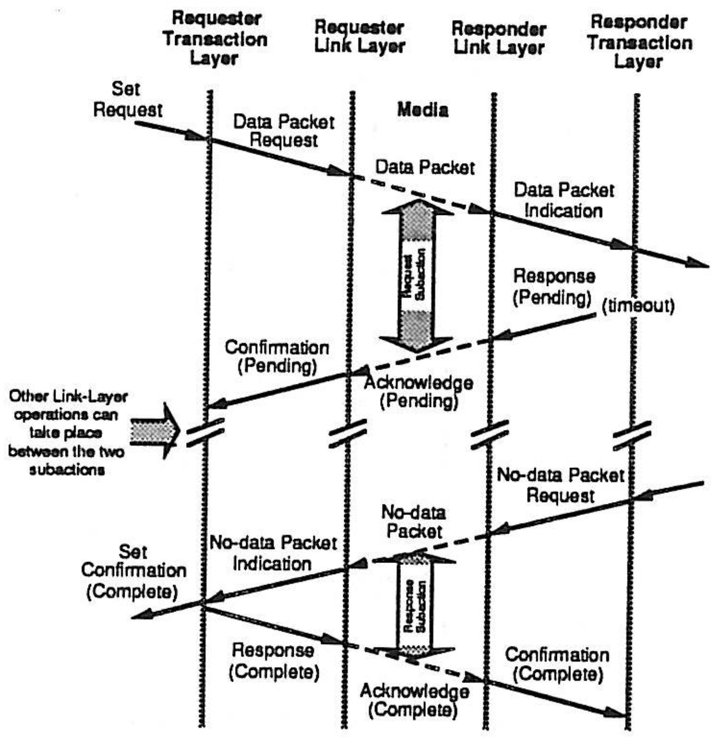


Figure 3-4. Split Transaction Example

When the Transaction Layer is busy and cannot accept a packet, the Link Layer sends a "reject" back to the requesting node. As an option, the Link Layer at the requesting node can retry the packet for a limited period. If bus retry traffic is high, then slower-acting nodes may be starved. To prevent this from happening there is a two-phase reject/retry protocol which allows all retries to be processed in a timely fashion.

3.1. Access Methods

To enhance the usefulness of Serial Bus, each node on the bus can use one of three different access methods. These access protocols are:

1. Fair Arbitration. Fair arbitration can be used by nodes which can equally share the bus bandwidth. This is the default method.
2. Urgent Arbitration. Urgent arbitration can be used by nodes which desire a majority of the bus bandwidth or have severe latency requirements. For example, a high-bandwidth real-time data collection node may use priority arbitration when critical data buffers are more than half full.
3. Isochronous Access. Data that is regularly generated and consumed, such as digital sound, can be transferred using an isochronous access method that guarantees bandwidth on the bus with very low overhead.

Urgent arbitration protocols are based on the node's priority. However, at least one half of the asynchronous subactions are always allocated fairly, which bounds the maximum time between bus arbitration and bus ownership. Even the lowest-priority node is never starved.

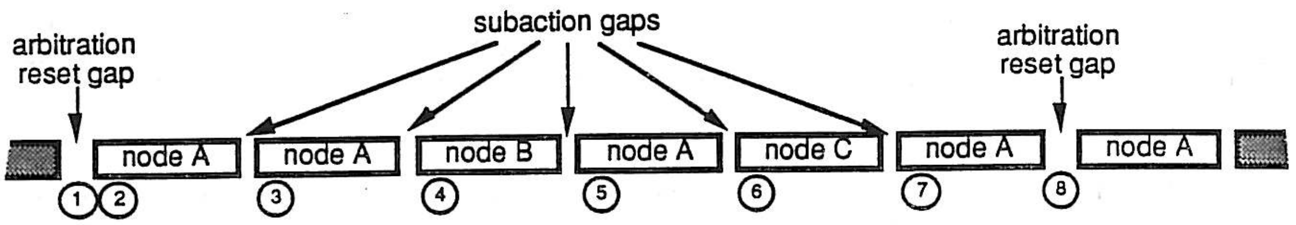
If a node wishes to send a transfer it first waits until the bus is idle for the appropriate period and then transmits its arbitration number, most-significant-bit first. Just before the end of each code-bit-period, the arbitrating nodes compare their arbitration bit to the value on the bus. As long as they are the same, arbitration continues. When they are different, arbitration has been lost; the node observing the difference immediately withdraws from the remainder of the arbitration process. The timing of the start of an arbitration sequence is critical: the maximum delay between valid bus state detection (subaction_gap, arbitration_reset_gap) and the assertion of the first arbitration symbol must be less than *41ns*.

3.1.1. Fair Arbitration

This arbitration method guarantees that only one node will still be transmitting by the end of the arbitration period. As described above, it only provides a strict priority access; the node with the highest arbitration number will always win. The Serial Bus adds a simple scheme that guarantees roughly half of the bus access opportunities to fair access and half for urgent access. This protocol depends on their being two gap lengths between subactions: a short "subaction gap" and a "fairness gap" which is somewhat longer. The modified method works as follows:

1) If the bus is idle for longer than the arbitration reset gap, a new "ownership interval" starts and all nodes set their "arbitration enable" flag.
2) All nodes have a "phase flag" that toggles after each transaction on the bus (even if the node is not participating). This flag is also cleared whenever the bus is idle longer than the arbitration reset gap.
3) A node using the fairness protocol may arbitrate for any transaction cycle as long as its arbitration enable flag is set. This flag is reset whenever the node wins an arbitration and does a transaction. This means that a fairness node can only do one transaction each ownership interval; i.e., it must wait until all other nodes using fairness protocols get a transaction in, at which point the bus will be idle long enough for an arbitration reset gap which ends the ownership interval and all nodes' arbitration enable flag is set.
4) A node using the urgent protocol may arbitrate for any transaction cycle as long as its arbitration enable flag is set OR its phase flag is set. Since the phase flag toggles each transaction, an urgent node that wants to continually use the bus will be restricted to using only half the bus access opportunities, plus whatever it would have gotten using the fairness protocol (even priority nodes can use the fairness protocol).

For example, assume that there are three nodes arbitrating for the bus with arbitration numbers 1 (node C), 2 (node B), and 3 (node A), and that B and C are fair nodes and A is an urgent node:



	Phase Bit (all nodes)	Enable Bit (node A)	Enable Bit (node B)	Enable Bit (node C)
(1) arbitration reset gap, fairness interval starts, all nodes clear phase, set enable	0	0	0	0
(2) fairness arbitrate, all arbitraion enable flags are set so the node with the highest address (node A) wins	1	1	0	0
(3) any arbitrate -- phase flag is 1, so node A can win again	0	1	0	0
(4) fairness arbitrate, arb enable for node A is clear, so node B wins	1	1	1	0
(5) any arbitrate -- phase flag is 1, so node A can win again	0	1	1	0
(6) fairness arbitrate, arbitration enable flags for nodes A and B are reset, so node C wins	1	1	1	1
(7) any arbitrate -- phase flag is 1, so node A can win again	0	1	1	1
(8) no node can arbitrate, phase bit and all arbitration enable flags are cleared, so the bus stays idle until an arbitration reset gap opens up, and a new fairness interval starts.	0	0	0	0

Figure 3-5. Arbitration example.

The timing for the gaps and arbitration bits is determined by the worst-case round-trip delay time for a signal on the medium. In particular, the unasserted portion of the arbitration bit time must be greater than the round-trip time plus the time necessary for a circuit to make a state decision.

For fair arbitration, the address has a minimal impact on the allocation of transmission opportunities. For urgent arbitration, the node with the largest priority value gets the majority of the bus transmission opportunities.

3.1.2. Extended Arbitration

Extended arbitration is used for the address selection operation of the bus management protocols. It always follows the fair arbitration algorithm, but it adds a set of bits to resolve the case of two nodes using the same test address during the an arbitration.

3.1.3. Isochronous Access

The basic arbitration protocol is quite adequate for nodes that do not require a guaranteed amount of bandwidth or a relatively precise timing reference (less than a microsecond, for instance). Some data, such as that related to digital sound or instrumentation, are more conveniently handled using an isochronous access method.

These isochronous services can be provided without upsetting the basic access protocol by establishing the convention that the highest priority arbitration number (all ones) is reserved for a "cycle" master that is responsible for maintaining a common clock source. The cycle master will try to transmit a special timing request called a "cycle start" at very specific intervals set by a "cycle synch" source (nominally 8 KHz ± 100 ppm, or 125 µsec ± 12.5 nsec). If another transfer is still taking place when the cycle synch occurs, then the cycle-start will be delayed resulting in significant jitter in the start time of the transfer. Since this jitter is frequently unacceptable for a node, the amount of time that the cycle start request was delayed is encoded within the request.

To simplify implementations, the cycle start request is encoded as a Transaction Layer quadlet set request with a local bus broadcast to the "cycle timer register". The cycle timer register is a 32-bit register in each node with isochronous service with lower half being a modulo 1536 counter which increments once each 82.5 nsec and the upper half a counter of the overflows from the lower half. The cycle master uses the cycle start set request to transfer its copy of the cycle timer register to all other isochronous nodes, keeping all nodes within a constant phase difference.

The response packets to the cycle-start request are isochronous "channels", with the ordering of the packets corresponding to the channel number, i.e., channel 1 is the first packet after the cycle-start, channel 2 is the second packet, etc. Nodes are assigned to channels by standard Set transactions sent by a manager application.

The second modification to the basic access protocol requires that all nodes cannot arbitrate for the bus until all the nodes that wish to transmit isochronous channels have done so. This is easily done by requiring a smaller minimum gap between channels than is needed for arbitration to start. The next figure illustrates the basic isochronous access system.

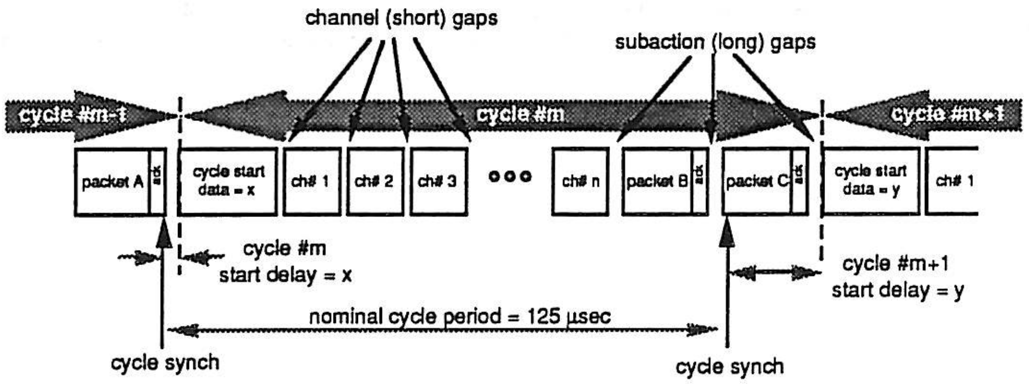


Figure 3-6. Cycle Structure

4. Physical Layer

The Physical Layer has two sublayers: the data encoding standards and the connector and physical interface standards.

4.1. Data Encoding

The requirements for the physical layer signal are:

1) During arbitration any node must receive a reliable "or" of the signals transmitted by all the nodes within one half of an arbitration bit time.

2) During packet transmission the signal must be sufficiently undistorted so that data can be reliably decoded at the chosen data rates.

There are two types of signals that can be present on the bus: arbitration and packet. Arbitration signals have the characteristics of long bit times and unreliable edges[1]. Packet signals are used whenever it is reliably known (via the protocol) that only a single node is driving the bus. The information signal is encoded in two different ways to match these characteristics, either a simple level-based signal for arbitration and acknowledgements, or with a relatively DC-free self-clocked block code for packet data.

4.1.1. Code Bit

Peer Physical Layer entities on the bus communicate via fixed-length code bits. Depending on the mode of operation, a code bit is either represented by an NRZI method ("1" is a transition, "0" is the absence of a transition) or an NRZ method ("1" is the high state, "0" is the low state). The NRZI method is used for packet symbols, NRZ for arbitration symbols. For the standard 49.152 Mbaud rate, each code bit is (1/49152000) seconds long (≈ 20.3 nsec).

4.1.2. Code Group

A code group is consecutive sequence of code bits and is used to represent a symbol on the medium. Implicit in the definition of code group is an establishment of code group boundaries by the Physical Layer. The two different types of symbols use different codes: a 4B5B block code for packet symbols, and a pulse-length modulated system for arbitration symbols.

4.1.3. Packet Data Coding

The packet symbol encoding system translates hexadecimal (4-bit) data symbols and special control symbols into 5 sequential NRZ code bits which are then further translated into an NRZI (transition) code. Decoding follows a reverse process, first translating NRZI code bits into NRZ code bits, and then into hexadecimal data or control symbols. The particular code is a modification of that used in the FDDI PHY document.

[1]During arbitration more than one node may be asserting the bus at the same time. Since the start times of the arbitration bits at the different nodes cannot be synchronized (and spatial differences result in significant skews in any case) the signal will only be reliable after a long settling time.

NRZI Code	Symbol	Meaning
11110	0	data symbol 0
01101	1	data symbol 1
10100	2	data symbol 2
10101	3	data symbol 3
01010	4	data symbol 4
01011	5	data symbol 5
01110	6	data symbol 6
01111	7	data symbol 7
10010	8	data symbol 8
10011	9	data symbol 9
10110	A	data symbol 10
10111	B	data symbol 11
11010	C	data symbol 12
11011	D	data symbol 13
11100	E	data symbol 14
11101	F	data symbol 15
11111	I	first three symbols of start delimiter
00100	H	third symbol of start delimiter
10001	T0	packet termination symbol 0
11001	T1	packet termination symbol 1
all others	V	violation

Table 4-1. Packet Data Symbol Table

The start delimiter of IIIH ("11111 11111 11111 00100") has two functions: first, to provide an edge-rich signal to start up the clock recovery, and second, to prevent ambiguities about the exact location of a code group boundary.

There are two different packet termination symbols, with T1 used when the ending bus state of the previous data symbol is asserted and T0 used when the ending bus state of the previous data symbol is unasserted. This guarantees that the last symbol consist of 3 asserted NRZ code bits and a single unasserted NRZ code bit.

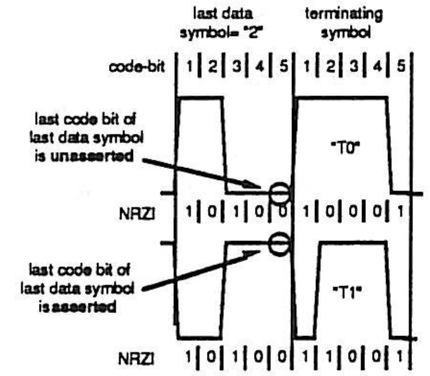


Figure 4-1. Termination Symbol

The V symbol is generated by the Physical layer whenever it receives an NRZI code that is not a data symbol or an I, H, or Tn.

The transmission speed will be 49.152 Mbaud (given the 4B5B modulation, this translates to 39.3216 megabits/second).

4.1.4. Arbitration Bit Encoding

The arbitration bit codes for both "1" and "0" have asserted periods so that long strings of "0" bits do not look like gaps between packets. The minimum period of each bit of arbitration is dependent on the length and signal propagation speed of the bus, with the primary requirement that the unasserted state of the "0" must be twice the bus round-trip transit time plus an amount for circuit delays. For the standard Serial Bus topology, this needs to be greater than *400* ns. To further simplify implementations, the arbitration bits are exact multiples of the 49.152 Mbaud packet code bits, so the unasserted period is *20* packet data code bits long = *407* ns. To keep the arbitration bits as short as possible, the "0" bit only has a short asserted period, *5* packet code bits long = *101* ns, resulting in a an arbitration bit period of *508* ns — so arbitration proceeds at roughly 2 megabits per second.

4.2. Physical Interface

The two environments for the Serial Bus, cable and backplane, have different physical interfaces. The backplane interface is specific to a particular backplane standard and includes the driver and receiver specifications as well as the particular signal lines used and their characteristics. There is only a single cable physical interface, and it has the following characteristics:

1) Up to four cables can be attached to a single node, but each node only has a single electrical interface. The cable interfaces are bussed together with the node interface as shown in the next figure.
2) Each cable provides a bus repeater function to propagate the signals between nodes, and there can be up to six cables separating any two nodes.
3) The electrical interface consists of two differential pairs carrying a low-voltage-swing signal in each direction and an additional pair of power conductors.
4) There can be a maximum of 6 cable hops between any two nodes (note that a backplane interface may count as one or two of these cable hops, depending on its speed).
5) The total cable length between any two nodes must be less than 10 meters.

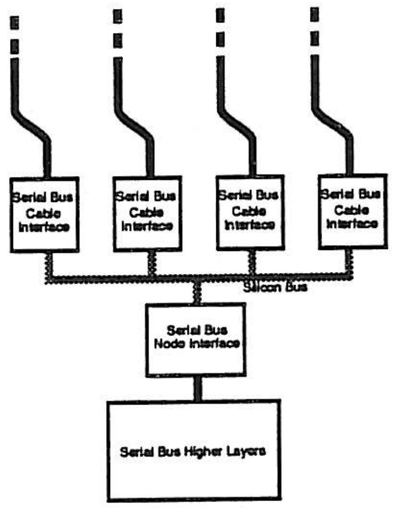

Serial Bus Cable Interface
Serial Bus Cable Interface
Serial Bus Cable Interface
Serial Bus Cable Interface
Silicon Bus
Serial Bus Node Interface
Serial Bus Higher Layers

Figure 4-2. Cable interface.

5. Formats

5.1. Subaction Formats

Subactions consist of an interpacket gap, an arbitration sequence (for asynchronous subactions), a packet, and an acknowledgement (for non-broadcast asynchronous subactions).

5.1.1. Interpacket Gap

The interpacket gap is an period of time during which the bus is unasserted. There are three types of gaps depending on the type of arbitration:

Iso_gap — Appears before channels (isochronous packets). The bus has been in a low (unasserted) state for at least *[500 ns]*.

Asynch_gap — Appears before asynchronous packets within a fairness interval. The bus has been in a low (unasserted) state for at least *[1 μs]*.

Arb_reset_gap — Appears before asynchronous packets when the fairness interval starts. The bus has been in a low (unasserted) state for at least *[1.5 μs]*.

5.1.2. Arbitration Sequence

The arbitration sequence is the bit-by-bit transmission and testing of an arbitration number using the algorithm outlined previously. The bits are encoded using the special arbitration bit code described later. This is ***not*** the same encoding used for the packets! There are two types of arbitration numbers: Standard and Extended.

5.1.2.1. Standard Arbitration Number

The standard arbitration number is used for all asynchronous packet transfers for normal data transfer. This field consists of a priority_class bit, an priority_number field, and a constant "one" bit, as illustrated below:

```
standard_arbitration_number format

            | p | priority_number | 1 |
              1          6          1
```

```
Where:
   p selects one of two priority classes, 0 = fair, 1 = urgent
   If p = 0, then priority_number is the node's offset_ID,
         otherwise it is a unique number establishing an access class of
         service
```

5.1.2.2. Extended Arbitration Number

The extended arbitration number is used for the address selection process. The extended_arbitration_number field consists of a constant "zero" bit, a test_address field, another constant "zero" bit, a random_number field, and a final contstant "one" bit, as illustrated below:

`extended_arbitration_number format`

0	test_address	0	random_number	1
1	6	1	6	1

```
Where:
   test_address is a tentative node offset_ID
   random_number is a 7-bit random number
```

The random number must change in a non-deterministic way each time the address selection process is run.

5.1.3. Packets

Packets are encoded using a 4B5B block code described in a later section. All packets begin with a starting delimiter and end with a termination symbol:

- Starting_delimiter — The starting delimiter provides the edge-rich signal that aids the startup of the clock recovery circuit in the Physical Layer and also provides a unique framing boundary. On transmission it is three I symbols followed by an H symbol. For reception only a single I followed by an H is required.
- Termination — The packet termination. It consists of either a T0 or a T1 symbol.

5.1.3.1. Asynchronous Packet

There are three fundamental types of asynchronous packets:

No-data packet:

sd	h	dest	src	tl	rt	rsv	hcrc	t
8	4	32	16	6	2	8	16	4

Quadlet packet:

sd	h	dest	src	tl	rt	rsv	qdata	hcrc	t
8	4	32	16	6	2	8	32	16	4

Long packet:

sd	h	dest	src	tl	rt	rsv	ecode	rsv	len	hcrc
8	4	32	16	6	2	8	8	12	12	16

--> cont.

cont. -->

qdata	pcrc	t
8*len	16	4

where:

sd	starting delimiter
h	packet header
dest	destination address
src	source address
tl	transaction label
rt	retry code
rsv	reserved (send as zero)
qdata	quadlet information
ecode	extended transaction code
len	number of bytes in block data (≤2048)
hcrc	header crc- (from sd to hcrc, exclusive)
bdata	block data
pcrc	packet crc (from sd to hcrc, exclusive, plus bdata)
t	termination symbol

5.1.3.2. Isochronous Packet

Isochronous (channel) packet:

sd	h	idata	icrc	t
8	4	8*n	16	4

where:

sd	starting delimiter
h	packet prefix = 0110
icrc	isochronous packet crc (from sd to icrc, exclusive)
idata	isochronous data

5.1.3.3. Packet Components

5.1.3.3.1. Packet Header

The packet header is a single 4-bit symbol and has the following format:

addr_size	tcode
0	3

where:
addr_size is 0 for 32-bit addresses and 1 for 64-bit address
tcode is the basic transaction code defined in 6.2.2.4

Tcode, the basic transaction code, is a 3-bit field carried in the 4-bit packet prefix defined in the Link Layer section:

000	quadlet set request
001	quadlet get request
010	quadlet set response
011	quadlet get response
100	(reserved)
101	extended request
110	isochronous packet
111	extended response

129

5.1.3.3.2. Destination Address (Request)

The addresses used by the Serial Bus Transaction Layer follow the P1212 standard. In particular, addresses are of two types: CSR and memory. The CSR addresses are the top 1/16th of the 32-bit address space and have the following mapping:

CSR_destination_address

1111	bus_number	offset_ID	register	00
4	10	6	10	2

```
where:
        bus_number    = 0 through 1022 = explicit bus address
                      = 1023 = local bus
        offset_ID     = 0 through 62 = explicit node number
                      = 63 = all nodes (broadcast)
        register      = CSR location
```

The only transactions that are guaranteed to work in CSR space are aligned quadlet get and set.

Memory addresses include all those in the lower 15/16ths of the 32-bit address space.

memory_destination_address

memory_address
32

```
where:
        memory_address      = 32-bit number smaller than 0xF0000000
```

5.1.3.3.3. Destination Address (Response)

The destination address for a packet has a different structure. It is always sent to CSR space, and the register field is replaced with the transaction label corresponding to the request and the response code.

`response_destination_address`

1111	bus_number	offset_ID	trans_label	rsv	r_code
4	10	6	6	2	4

Where:

trans_label transaction label of the corresponding request

rsv reserved, transmitted as zeros but make no assumptions on reception

r_code response code:

_correct	x000	Completion successful, normal operation
_advise	x001	Completion successful, abnormal operation
(reserved)	x010	(reserved error condition)
_size	x011	Transaction size limit exceeded
_conflict	x100	Conflict
_data	x101	Data error, bad CRC on data
_type	x110	Unsupported transaction
_address	x111	Addressing error
responder_	0xxx	Responder provided status
agent_	1xxx	Bridge provided status

5.1.3.3.4. Source Address

The source_address is the node_ID of the source of the packet. This is simply the concatenation of the bus_number and offset_ID fields.

bus_number	offset_ID
10	6

/31

5.1.3.3.5. Transaction Label & Retry Codes

The transaction label and retry code follow the source_address in a packet and have the following format:

transaction_label	retry_code
6	2

Where:

transaction_label	In a request, this is a unique ID for transactions pending from a particular requesting node. A node may not have more than one transaction pending for each transaction label. In a response this field is **reserved.**
retry_code	Identifier for the transaction retry protocol:

00	initial transaction
01	(reserved)
10	retry-A
11	retry-B

5.1.3.3.6. Extended Transaction Code

The extended transaction code is 32 bits and is located in the "qdata" field of the packet used in any extended request or response:

ettype	sub_cmd	reserved	length
4	4	12	12

Where:

ettype — extended transaction type

0000	set
0001	get
0010	lock
0011	reserved
x1xx	reserved
1xxx	reserved

sub_cmd — lock sub-command[1] (field is reserved if ettype ≠ lock)

0000	(reserved)
0001	mask_swap
0010	compare_swap
0011	fetch_add
0100	(reserved)
0101	bounded_add
0110	wrap_add
0111	(vendor dependent)
1xxx	(reserved)

length — the number of bytes in the block data field (≤ 2048)

[1]These operations are defined in detail in the P1212 specification.

132

5.1.3.3.7. CRC's

All CRC's use the CCITT V.41 equation: $X^{16} + X^{12} + X^5 + 1$. The CRC generator is preset to zeros.

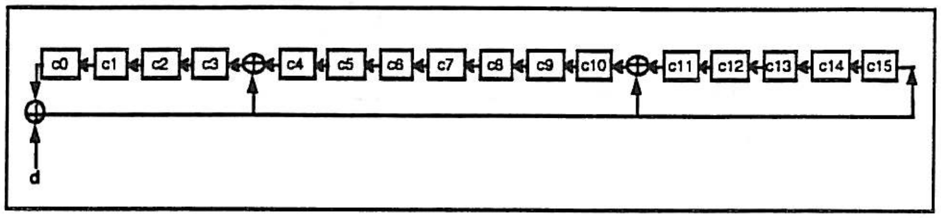


Figure 5-1. 16-bit CRC generator.

5.1.3.4. Acknowledge

The acknowledge is a base-one code of arbitration "1" symbols (i.e., the acknowledge is 1, 2, 3, or 4 arbitration "1" symbols):

Code	Number of arb "1" symbols	Meaning
missing	0	No node at the destination addresses <u>or</u> CRC error on the packet header.
complete	1	The packet was accepted and the destination Transaction Layer completed the transaction. A "complete" response is sent to the source Transaction Layer.
pending	2	For requests: the packet was accepted and forwarded to the destination Transaction Layer. The destination Transaction Layer will generate a response later.
		For responses: there was an error in the data field of the response packet.
reject_A	3	The packet could not be accepted. The destination Transaction Layer was busy and will not send a response.
reject_B	4	The packet could not be accepted. The destination Transaction Layer was busy and will not send a response.

The acknowledge can be preceded by gap no greater than an iso_gap. The two types of rejects: reject_A and reject_B, are used by the Transaction Layer to allow fair access when retries are necessary.

5.1.3.4.1. Acknowledge Timing

Acknowledge timing is equivalent to arbitration timing, with the timing trigger being the end of the data packet:

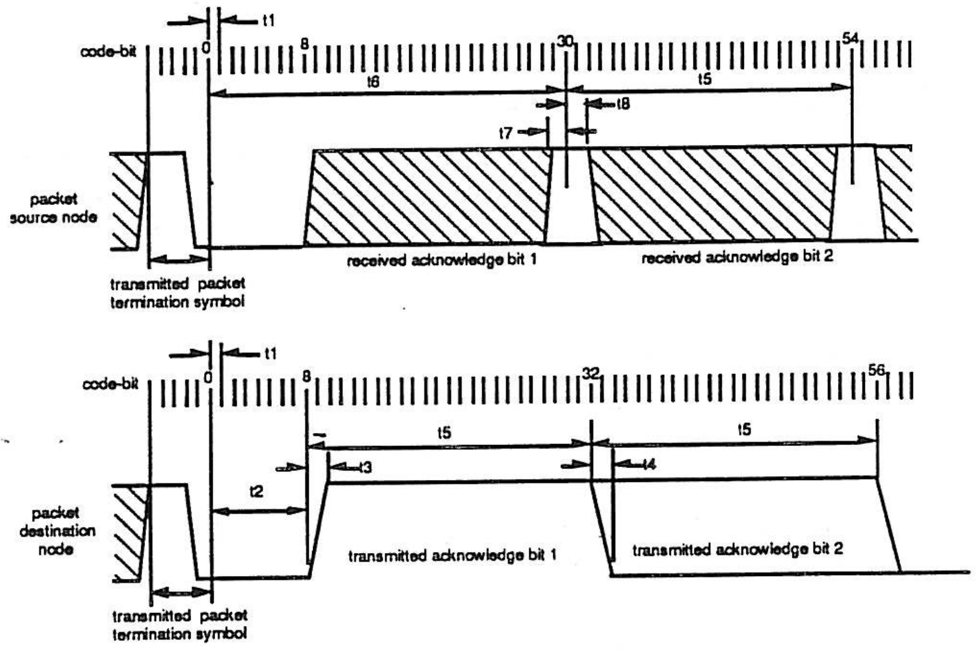

t1
code-bit
0
8
30
54
t6
t5
t8
t7
packet source node
received acknowledge bit 1
received acknowledge bit 2
transmitted packet termination symbol
t1
code-bit
0
8
32
56
t5
t5
t3
t4
t2
packet destination node
transmitted acknowledge bit 1
transmitted acknowledge bit 2
transmitted packet termination symbol

Figure 5-1. Acknowledge timing

t1	code bit time	1/(49.152 MHz ± 100ppm)
t2	first acknowledge symbol delay	8 * t1
t3	rise time	*10 ns max*
t4	fall time	*20 ns max*
t5	acknowledge symbol period	24 * t1
t6	sampling delay (1st bit)	30 * t1
t7	setup time	*10 ns min*
t8	hold time	*0 ns min*

5.2. Transaction Packet Formats

5.2.1. Set Request

5.2.1.1. Quadlet

Quadlet packet:

sd	h	dest	src	tl	rt	rsv	qdata	hcrc	t
8	4	32	16	6	2	8	32	16	4

h = 0000

134

5.2.1.2. Block

Long packet:

sd	h	dest	src	tl	rt	rsv	ecode	rsv	len	hcrc	--> cont.
8	4	32	16	6	2	8	8	12	12	16	

cont. -->	qdata	pcrc	t
	8*len	16	4

h = 0101, ecode = 0000 0000

5.2.2. Set Response

No-data packet:

sd	h	dest	src	tl	rt	rsv	hcrc	t
8	4	32	16	6	2	8	16	4

h = 0010

5.2.3. Get Request

5.2.3.1. Quadlet

No-data packet:

sd	h	dest	src	tl	rt	rsv	hcrc	t
8	4	32	16	6	2	8	16	4

h = 0001

5.2.3.2. Block

Long packet:

sd	h	dest	src	tl	rt	rsv	ecode	rsv	len	hcrc	t
8	4	32	16	6	2	8	8	12	12	16	4

h = 0101, ecode = 0001 0000

5.2.4. Get Response

5.2.4.1. Quadlet

Quadlet packet:

sd	h	dest	src	tl	rt	rsv	qdata	hcrc	t
8	4	32	16	6	2	8	32	16	4

h = 0011

5.2.4.2. Block

Long packet:

sd	h	dest	src	tl	rt	rsv	ecode	rsv	len	hcrc	--> cont.
8	4	32	16	6	2	8	8	12	12	16	

cont. -->	qdata	pcrc	t
	8*len	16	4

h = 0111, ecode = 0001 0000

5.2.5. Lock Request

Long packet:

sd	h	dest	src	tl	rt	rsv	ecode	rsv	len = 4	hcrc	--> cont.
8	4	32	16	6	2	8	8	12	12	16	

cont. -->	qdata	pcrc	t
	32	16	4

h = 0101, ec oce = 0010 mmmm, where "mmmm" is the lock operation code

5.2.6. Lock Response

Long packet:

sd	h	dest	src	tl	rt	rsv	ecode	rsv	len = 4	hcrc	--> con
8	4	32	16	6	2	8	8	12	12	16	

cont. -->	qdata	pcrc	t
	32	16	4

h = 0111, ec oce = 0010 mmmm, where "mmmm" is the lock operation code

5.2.7. Cycle Start Request

The Cycle Start request is just a particular form of the quadlet set request.

Quadlet packet:

sd	h	dest	src	tl	rt	rsv	qdata	hcrc	t
8	4	32	16	6	2	8	32	16	4

h = 0000 (Quadlet Set Request), dest = 0xffff f200 (Broadcast to all nodes' Cycle Timer Register)

6. Bus Management

The Bus Management protocol provides a multitude of management facilities:

1) Arbitration number/address assignment. It is unacceptable to require the user to manually set the address of a node. Instead, the Serial Bus protocol defines a special address arbitration mechanism that allows nodes to chose a unique address dynamically.

2) Cycle master arbitration. Before any cycle transactions can start, all candidate cycle master nodes must arbitrate which will actually become the cycle master. The candidate with the most accurate clock should become master. Most of the time, there will only be a single cycle master candidate.

3) Isochronous channel assignment.

4) Error control. The basic philosophy for error handling is to note that there is an error, but to not necessarily retry any transfer (retries are reserved for busy conditions). The Serial Bus itself does not guarantee delivery: that is up to higher layers.

7. Performance

The performance of the Serial Bus varies according to two primary parameters: data packet bit rate and bus round-trip time. For the P1394 standard the bus round trip time is limited to 25 / 49.152 MHz ≈ 0.51 μsec and the data packet bit rate is either 4 /(5 * 49.152 MHz) ≈ 39 Mbit/sec or 4/(4 * 196.61 MHz) ≈ 157 Mbit/sec. The true carrying capacity of the bus varies in a complex way with the offered traffic since the per-packet overhead is largely fixed.

7.1. Asynchronous

The per-subaction overhead for asychronous data is:

fixed asynch overhead = intertransaction gap + arbitration+ fixed part of packet + ack gap + ack

The intertransaction minimum is 1.5 μsec, the normal arbitration takes 8 * 0.5 = 4 μsec, and the acknowledge gap is about 0.5 μsec. The acknowledge is either 0.5 μsec for the "complete" ack or 1.0 μsec for the "pending" ack, while the fixed part of a block packet (≠ 4 byte) is 152 bits long (starting delimitor, transaction code, addresses, label, length field, header and data CRCs, and termination). A quadlet subaction is fixed size at 136 bits. Putting all this in the formula above we get:

Subaction type	**49.15 Mbaud**	**196.61 Mbaud**	
quadlet subaction	10.04	7.45	μsec
block subaction fixed overhead	10.45	7.55	μsec
256 byte subaction	62.54	20.57	μsec
1024 byte subaction	218.79	59.64	μsec

Although a unified transaction "set" is the same as a subaction, "get" transactions have the additional fixed overhead of the request subaction:

get fixed overhead = request subaction + response subaction fixed overhead

For a quadlet transaction, the request is a no-data subaction and the response is a quadlet subaction. For a block transaction, the request is a quadlet subaction and the response is a block subaction, giving us:

Get transaction type	**49.15 Mbaud**	**196.61 Mbaud**	
get quadlet (4 byte)	19.28	14.70	μsec
get block (≠4 byte) fixed overhead	20.50	15.00	μsec
256 byte get	72.58	28.02	μsec
1024 byte get	228.83	67.09	μsec

The actual data throughput for transactions will vary with the number of bytes transferred, since smaller packets have a higher percentage used for overhead. The throughput can be calculated using the following equation:

$$\text{actual data throughput} = \frac{\text{\# of data bits per transaction}}{\text{transaction time}}$$

The following table contains example throughput numbers for several interesting cases:

Transaction	**49.15 Mbaud**	**196.61 Mbaud**	
quadlet set	3.19	4.29	Bits/sec
256-byte set	32.75	99.55	Bits/sec
1024-byte set	37.44	137.37	Bits/sec
quadlet get	1.62	2.10	Bits/sec
256-byte get	28.02	71.78	Bits/sec
1024-byte get	35.72	121.19	Bits/sec

7.2. Isochronous

The per-packet overhead for isochronous data is:

fixed iso overhead = interchannel gap + fixed part of channel

The interchannel gap is no larger than 1.0 μsec while the fixed part of the channel is 40 bits long (starting delimitor, transaction code, CRC, and termination). This gives a channel overhead time of 1.5 μsec at 49 Mbaud and 0.8 μsec at 196 Mbaud.

Channel size	**49.15 Mbaud**	**196.61 Mbaud**	
0 byte	1.53	0.76	μsec
1 byte (ISDN B channel 64 kbit/sec)	1.73	0.81	μsec
24 byte (DAT stereo pair, T1 channel, 1.536 Mbit/sec)	7.02	2.14	μsec

The overall capacity of the isochronous transport is limited to the number of channels that can be sent in 125 μsec (less the cycle start overhead). The cycle start transaction is a normal quadlet set (always at 49 Mbaud), so the following inequality must be met for correct isochronous operation.

sum of all channels $\leq$ 125 μsec - quadlet set = 115 μsec

This allows the following example isochronous traffic mixes:

Channel size	**49.15 Mbaud**	**196.61 Mbaud**	
1 byte (ISDN B channels)	69	147	channels
24 byte (DAT stereo pairs)	18	60	channels

139

High Speed Serial Bus

Project 1394 of the
Microcomputer Standards Committee
of the IEEE Computer Society

Michael Teener
Chair, P1394 Working Group
Apple Computer, Inc.
3535 Monroe St, MS 60-I
Santa Clara, CA 95051
408-974-3521
teener@apple.com

Goals

- Alternate Bus
  - Bridging between different parallel busses
  - Redundant path for configuration and maintenance
  - For extremely low cost modules
- Low cost peripheral bus
- Bus bridge
- Compatible architecture with other IEEE 32-bit busses
  - Follow proposed P1212 CSR standard (control and status register)

Goals (cont.)

- Modest cost
  - < $15 for cable, socket & interface ICs (very large volume)
- High bandwidth
  - Fast as possible given cost goals
  - Lower CPU overhead via simpler protocols & DMA
  - Performance extensibility
- Add isochronous service
  - from 64 kbps for voice to 1.5 Mbps for stereo HiFi sound to >20 Mbps for compressed video

"Isochronous" ??

- Iso (same) chronous (time) :
  - – Uniform in time
  - – Having equal duration
  - – Recurring at regular intervals
- Examples
  (source and destination for real-time data)

ISDN	8 kHz x 8 bits	64 kbps
CD	44.1 kHz x 16 bits x 2 channels	1.4 Mbps
DAT	48 kHz x 16 bits x 2 channels	1.5 Mbps
Video	25 or 30 fps	variable

140

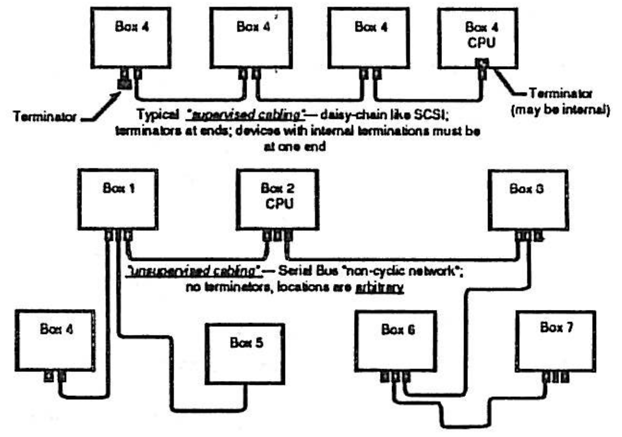
Goals for External Cable

P1394 High Speed Serial Bus

- **Reduce EMC problems**
  - Low voltage swing, low current, low skew differential signal.
  - Shielding & isolation designed in from the start.
- **Improved ergonomics**
  - Small, flexible cable
  - Robust small connectors with self-locking feature
  - No terminators, few topology rules

Unsupervised!

P1394 High Speed Serial Bus


Box 4
Box 4
Box 4
Box 4 CPU
Terminator
Terminator (may be internal)
Typical *"supervised cabling"*— daisy-chain like SCSI; terminators at ends; devices with internal terminations must be at one end
Box 1
Box 2 CPU
Box 3
"unsupervised cabling"— Serial Bus "non-cyclic network"; no terminators, locations are *arbitrary*
Box 4
Box 5
Box 6
Box 7

Protocols

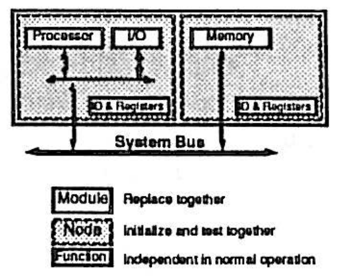
P1394 High Speed Serial Bus

- **P1394 High Speed Serial Bus**
  - "Bus-like" logical architecture
- **P1212 CSR Architecture**
  - Standardized addressing
  - Well-defined control and status registers
  - Standardized transactions

P1212 Module Architecture

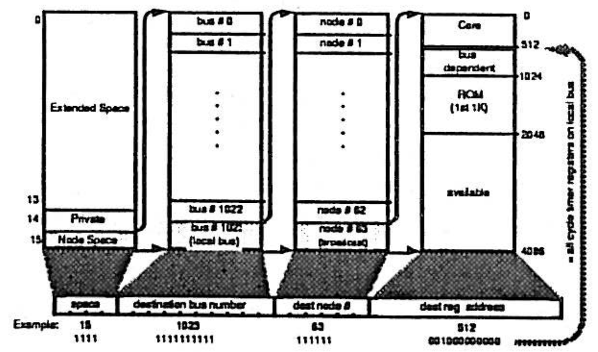
P1394 High Speed Serial Bus

Module Architecture



P1212 Addressing

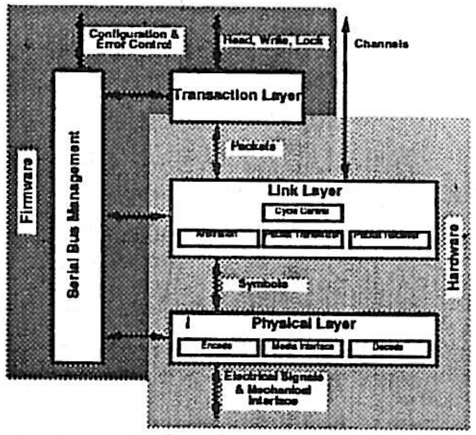
P1394
High Speed
Serial Bus



Apple Computer, Inc. mdt 1/91, slide 9

Protocol Stack

P1394
High Speed
Serial Bus

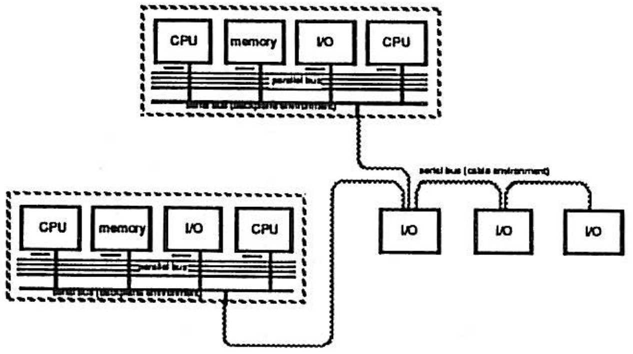


Apple Computer, Inc. mdt 1/91, slide 10

141

Physical Layer Topology

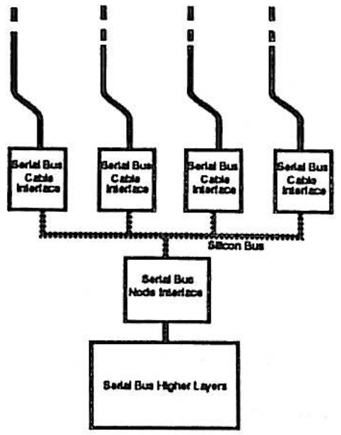
P1394
High Speed
Serial Bus


CPU
memory
I/O
CPU
parallel bus
serial bus (backplane environment)
serial bus (cable environment)
I/O
I/O
I/O

Apple Computer, Inc. mdt 1/91, slide 11

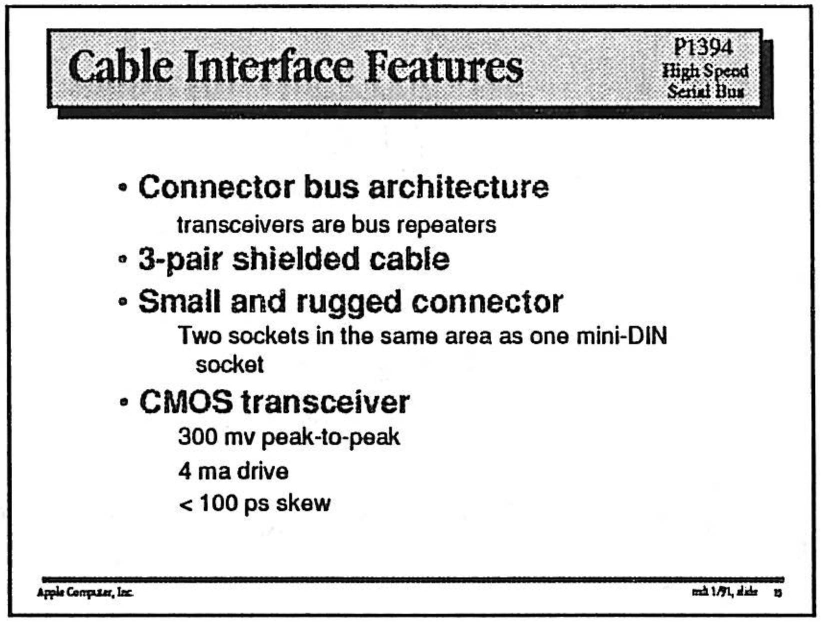
Cable Interface

P1394
High Speed
Serial Bus


Serial Bus Cable Interface
Serial Bus Cable Interface
Serial Bus Cable Interface
Serial Bus Cable Interface
Silicon Bus
Serial Bus Node Interface
Serial Bus Higher Layers

- Silicon bus provides "or"ing of signal
- Cables and transceivers are bus repeaters
- Limit of 6 cable hops and 10 meters of cable between any two nodes

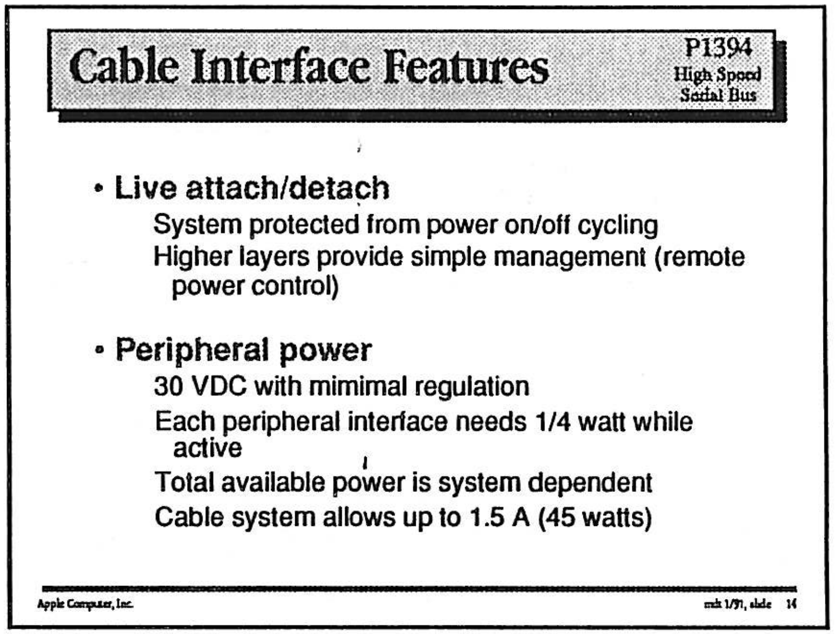
Apple Computer, Inc. mdt 1/91, slide 12



Cable Interface Features

P1394
High Speed
Serial Bus

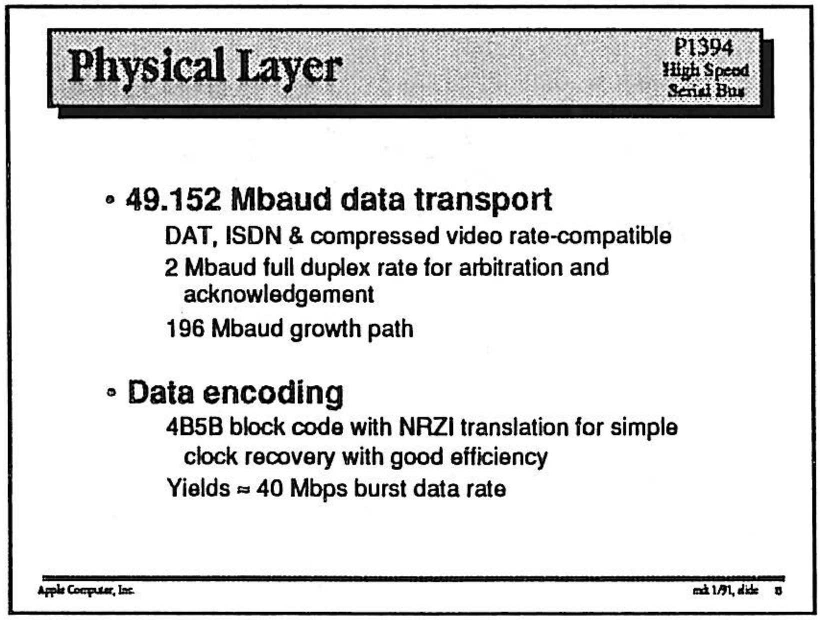
- **Connector bus architecture**
  transceivers are bus repeaters
- **3-pair shielded cable**
- **Small and rugged connector**
  Two sockets in the same area as one mini-DIN socket
- **CMOS transceiver**
  300 mv peak-to-peak
  4 ma drive
  < 100 ps skew



Cable Interface Features

P1394
High Speed
Serial Bus

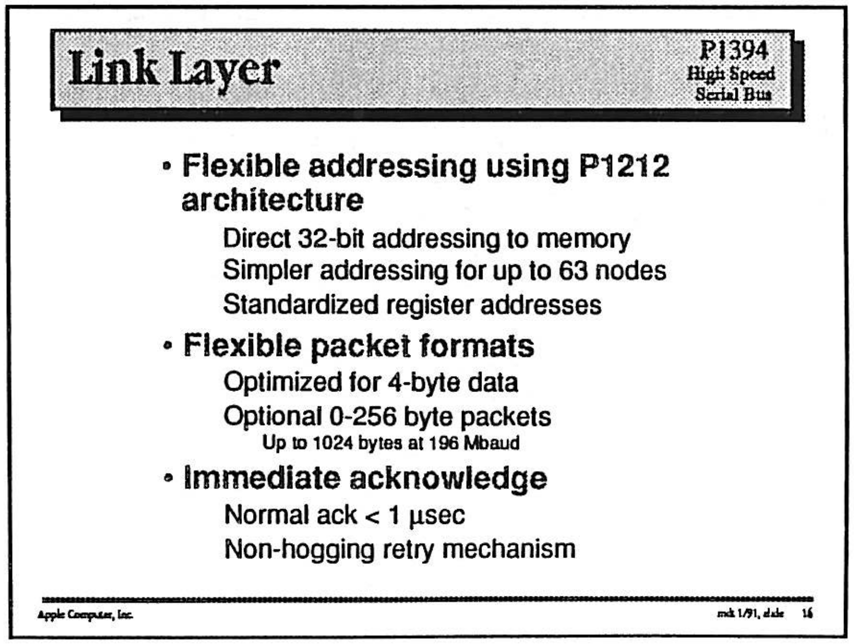
- **Live attach/detach**
  System protected from power on/off cycling
  Higher layers provide simple management (remote power control)
- **Peripheral power**
  30 VDC with mimimal regulation
  Each peripheral interface needs 1/4 watt while active
  Total available power is system dependent
  Cable system allows up to 1.5 A (45 watts)



Physical Layer

P1394
High Speed
Serial Bus

- **49.152 Mbaud data transport**
  DAT, ISDN & compressed video rate-compatible
  2 Mbaud full duplex rate for arbitration and acknowledgement
  196 Mbaud growth path
- **Data encoding**
  4B5B block code with NRZI translation for simple clock recovery with good efficiency
  Yields ≈ 40 Mbps burst data rate



Link Layer

P1394
High Speed
Serial Bus

- **Flexible addressing using P1212 architecture**
  Direct 32-bit addressing to memory
  Simpler addressing for up to 63 nodes
  Standardized register addresses
- **Flexible packet formats**
  Optimized for 4-byte data
  Optional 0-256 byte packets
  Up to 1024 bytes at 196 Mbaud
- **Immediate acknowledge**
  Normal ack < 1 μsec
  Non-hogging retry mechanism

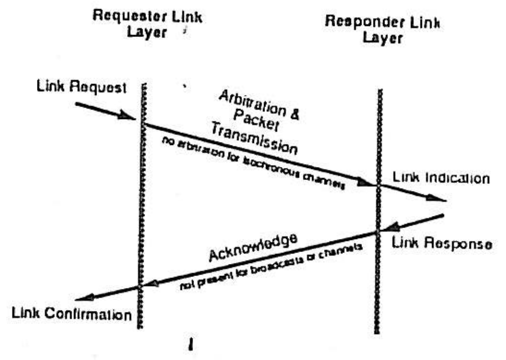
Link Layer (cont.)

P1394 High Speed Serial Bus

- **Fair and priority access**
  - Bit-serial arbitration
  - Automatic assignment of addresses
  - Access opportunities split between fair and priority modes
- **Isochronous transport**
  - Optional
  - Up to 32 "channels" each 125 µsec period
  - 0-256 bytes in each channel
  - 82 ns max jitter in 125 µsec clock

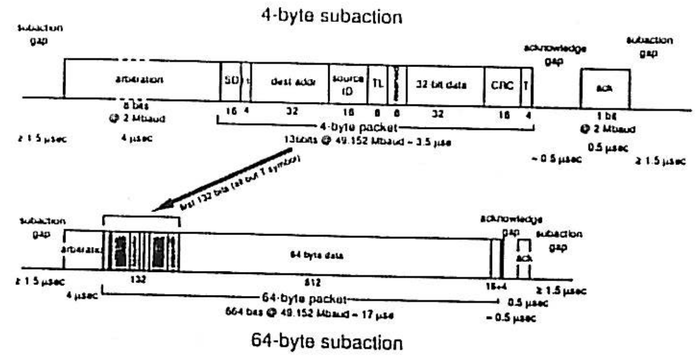
Link Layer Operation

P1394 High Speed Serial Bus


Requester Link Layer
Responder Link Layer
Link Request
Arbitration & Packet Transmission
no arbitration for isochronous channels
Link Indication
Acknowledge
not present for broadcasts or channels
Link Response
Link Confirmation

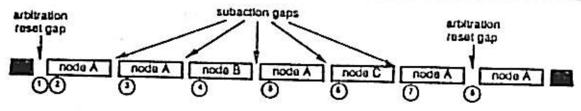
Example Packets

P1394 High Speed Serial Bus



Fair and Priority Arbitration

P1394 High Speed Serial Bus

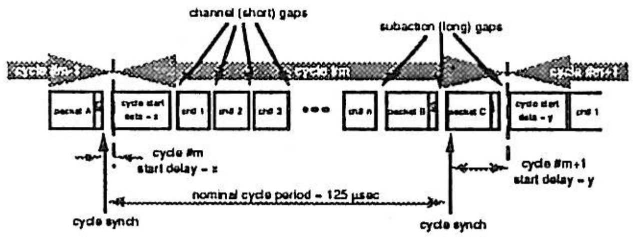


	Phase Bit (all nodes)	Enable Bit (node A)	Enable Bit (node B)	Enable Bit (node C)
(1) arbitration reset gap, fairness interval starts, all nodes clear phase, set enable	0	0	0	0
(2) fairness arbitrate, all arbitraion enable flags are set so the node with the highest address (node A) wins	1	1	0	0
(3) any arbitrate – phase flag is 1, so node A can win again	0	1	0	0
(4) fairness arbitrate, arb enable for node A is clear, so node B wins	1	1	1	0
(5) any arbitrate – phase flag is 1, so node A can win again	0	1	1	0
(6) fairness arbitrate, arbitration enable flags for nodes A and B are reset, so node C wins	1	1	1	1
(7) any arbitrate – phase flag is 1, so node A can win again	0	1	1	1
(8) no node can arbitrate, phase bit and all arbitration enable flags are cleared, so the bus stays idle until an arbitration reset gap opens up, and a new fairness interval starts.	0	0	0	0

144

Cycle Structure

P1394 High Speed Serial Bus



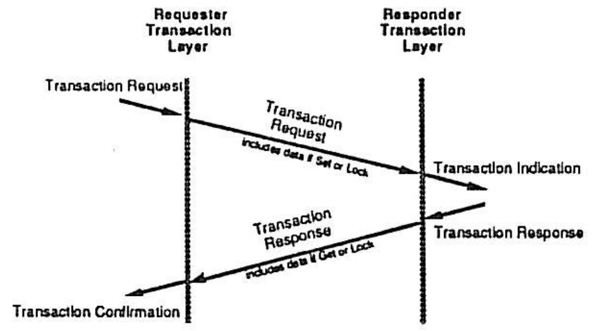
Transaction Layer

P1394 High Speed Serial Bus

- **Multiple transaction types**
  - Optimized for 4-byte operations
  - Read and write quadlet are required
  - 0-256 byte operations optional
  - Lock transactions optional
    - Compare-and-swap
- **Advanced management**
  - Automatic address assignment (no switches)
  - Standardized addresses
    - command, status, interrupt, etc.
  - Configuration ROM
    - device type, name, manufacturer, power requirements, capabilities, driver code

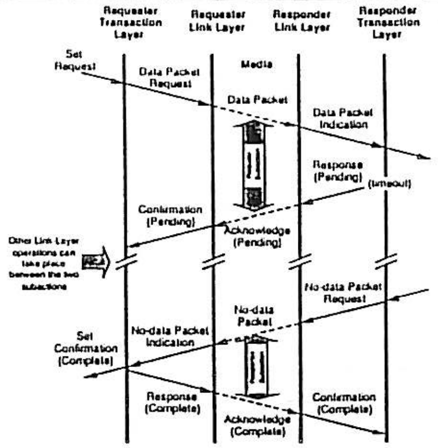
Transaction Layer Operation

P1394 High Speed Serial Bus



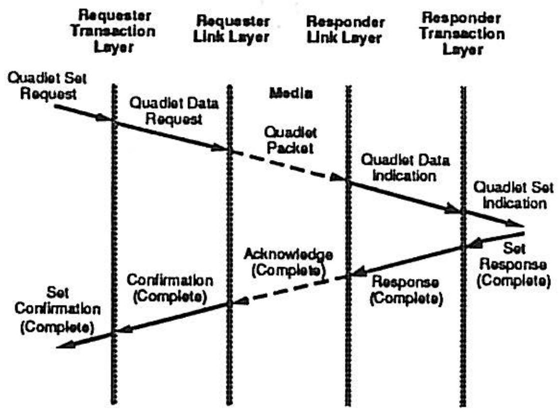
Split Transaction

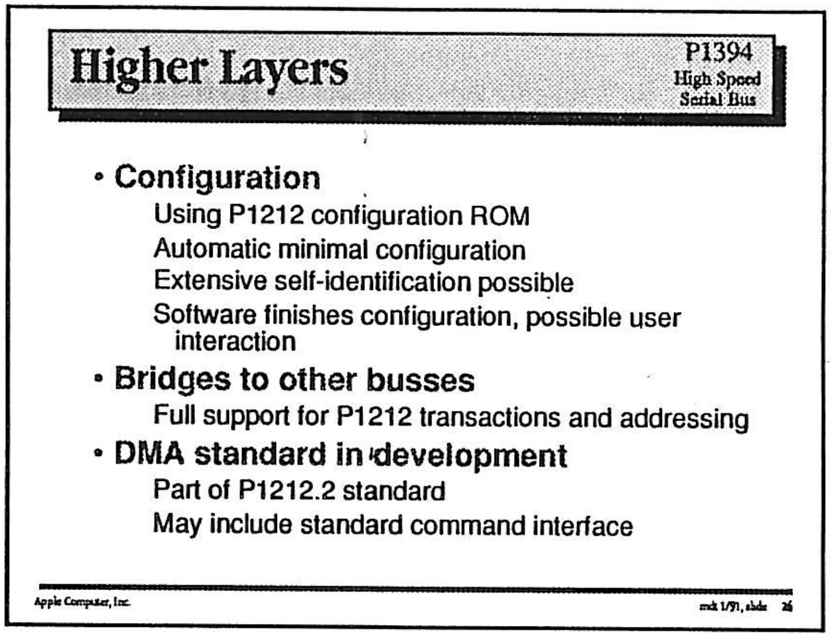
P1394 High Speed Serial Bus



Fast Transaction

P1394
High Speed
Serial Bus



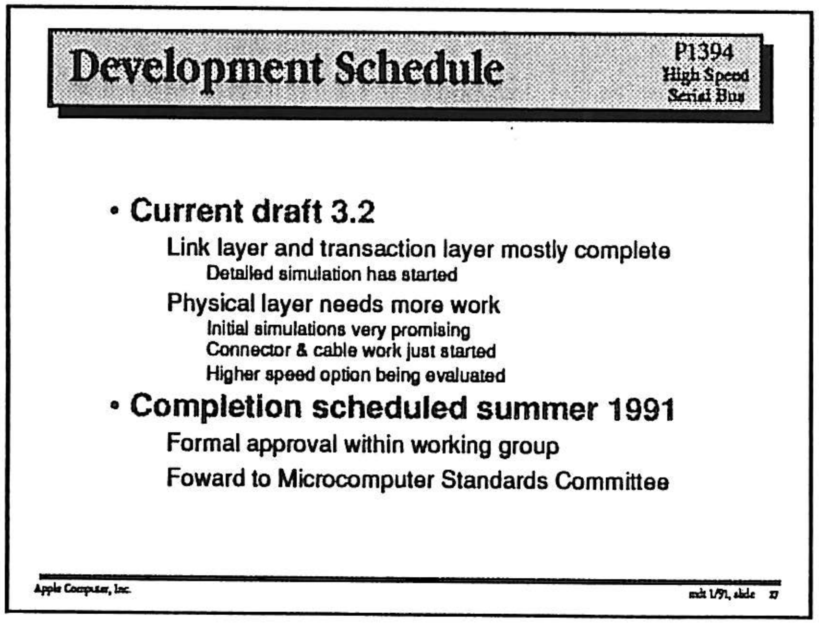


Higher Layers

P1394
High Speed
Serial Bus

- Configuration
  - Using P1212 configuration ROM
  - Automatic minimal configuration
  - Extensive self-identification possible
  - Software finishes configuration, possible user interaction
- Bridges to other busses
  - Full support for P1212 transactions and addressing
- DMA standard in development
  - Part of P1212.2 standard
  - May include standard command interface

145



Development Schedule

P1394
High Speed
Serial Bus

- Current draft 3.2
  - Link layer and transaction layer mostly complete
    - Detailed simulation has started
  - Physical layer needs more work
    - Initial simulations very promising
    - Connector & cable work just started
    - Higher speed option being evaluated
- Completion scheduled summer 1991
  - Formal approval within working group
  - Foward to Microcomputer Standards Committee