Annex I. P1394 Link-Phy Interface Specification

I.1 Document Scope

This document describes a Link-Phy interface for the proposed 1394 high speed serial bus standard. It covers the protocol, signal timing and galvanic isolation circuitry. It does not cover specific operation of the phy except for behavior with respect to this interface.

In this document, bit 0 of any multiple-bit bus is most significant and transmitted first on the serial bus. The proposed data rates which are referred Mbit/secto in multiples of 100 Mbit/secare actually multiples of 98.304 Mbit/sec. 100 Mbit/secterms are used to save everyone's chops. These rates are the actual 'bit' rates, independent of the encoding scheme. The actual clock rate in a redundant encoding scheme is referred to as a 'baud' rate, and is independent of the clock rate of this interface. In the timing diagrams in this document, each bit cell represents one clock sample time. The specific clock to data timing relationships are described in the 'AC Timing' section.

The interface described here supports data rates of 100, 200, and 400 Mbit/sec, and can be extended beyond 400 Mbit/sec if that need ever arises.

I.2 Overview

The interface described in this document is a scalable, cost-effective method to connect one 1394 link chip to one 1394 phy chip. The width of the data bus scales with the highest speed both chips can support, using two pins per 100 Mbit/sec. The clock rate of the signals at this interface remains constant, independent of speed, to support galvanic isolation for implementations where it is desirable.

The phy has control over the bidirectional pins. The link only drives these pins when control is transferred to it by the phy. The link performs all unsolicited activity through a dedicated request pin. The possible actions which may occur on the interface are categorized as transmit, receive, status, and request. These actions are described in detail later in this document.

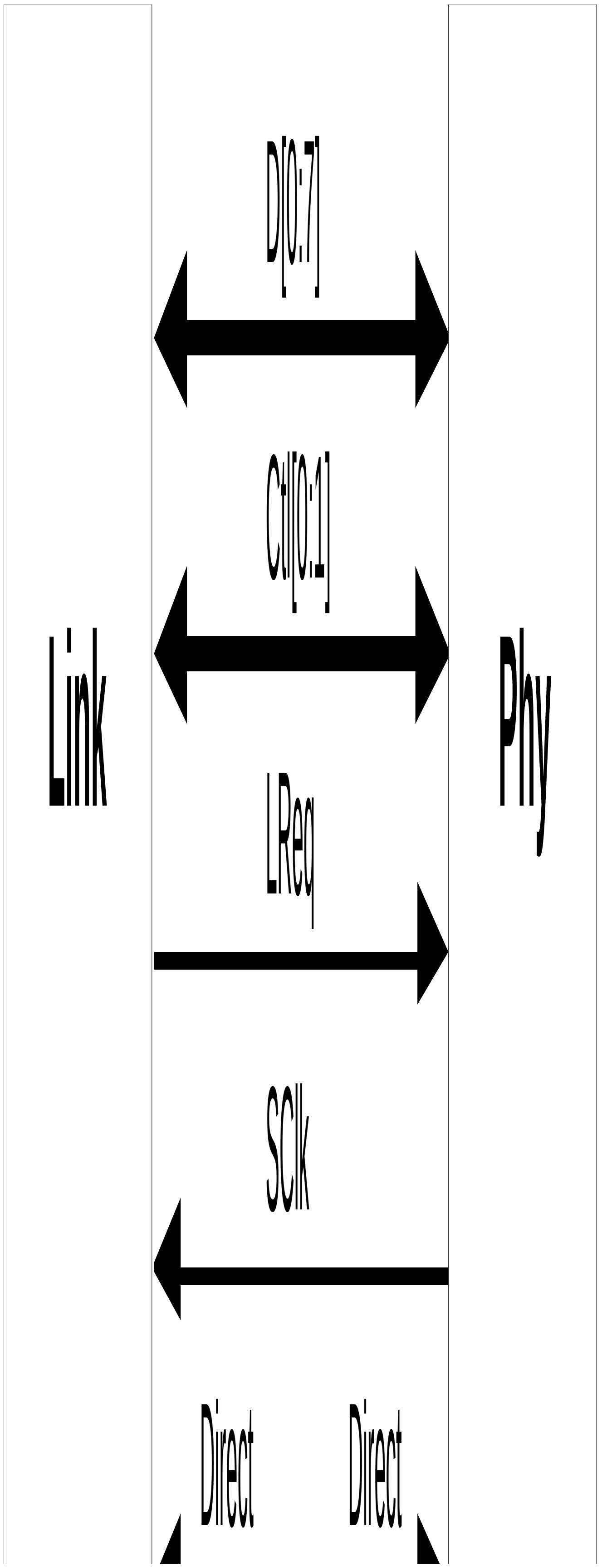
D[0:7]
Ctl[0:1]
LReq
SClk
Direct
Direct
Link
Phy

Figure I-1 – PHY-Link interface

Table I-1 – Pin description

Name	**Driven by**	**Description**
D[0:7]	both	Data
Ctl[0:1]	both	Control
LReq	link	Link Request port
SClk	phy	49.152 MHz (sync to serial bus) clock
Direct	neither	Controls differentiator for interface pins

Data is carried between the two chips on the D bus. The width of the D bus depends on the maximum speed of the connected phy chip, 2 bits per 100 Mbit/sec. In multiple speed implementations, the portion of the D bus which carries packet data is left-justified in the D bus (starting with bit 0). The Ctl bus carries control information and is always 2 bits wide. The LReq pin is used by the link to request access to the serial bus and to read or write phy registers. The Direct pin is used to disable the digital differentiator on the D, Ctl, and LReq pins, indicating that the two chips are directly connected, rather than through an isolation barrier.

Whenever control is transferred between the phy and the link, the side giving up control always drives the cxontrol and data pins to logic 0 levels for one clock before tristating its output buffers. This is necessary to ensure that the differentiator circuit can operated properly. This procedure is built into the operational descriptions and timing diagrams that follow.

I.3 Operation

There are four basic operations which may occur in the interface: request, status, transmit, and receive . All but request are initiated by the phy. The link uses the request operation to read or write an internal phy register or to ask the phy to initiate a transmit action. The phy initiates a receive action whenever a packet is received from the serial bus.

The Ctl bus is always 2 bits wide, independent of speed. The encoding of these pins is as follows::

Table I-2 – Ctl[0:1] when phy is driving

Ctl[0:1]	**Name**	**Meaning**
00	Idle	No activity
01	Status	The phy is sending status information to the link
10	Receive	An incoming packet is being transferred from the phy to the link
11	Transmit	The link is granted the bus to send a packet

Table I-3 – CTL[0:1] when the link is driving (upon a grant from the phy)

Ctl[0:1]	**Name**	**Meaning**
00	Idle	Transmission complete, release bus
01	Hold	The link is holding the bus while preparing data or indicating that it wishes to reacquire the bus without arbitrating to send another packet.
10	Transmit	The link is sending a packet to the phy
11	unused	unused

The use of these signals is described in the sections that follow.

I.3.1 Request

To request the bus or access a phy register, the link sends a short serial stream to the phy on the LReq pin. The information sent includes the type of request or the speed at which the packet is to be sent, or a read or write command. The transfer can be either 7 bits, 9 bits, or 17 bits, depending on whether it is a bus request, a read access, or a write access, respectively. A stop bit of 0 is required after each type of request transfer before another transfer may begin.

The timing for this pin and the definition of the bits in the transfer are shown below.

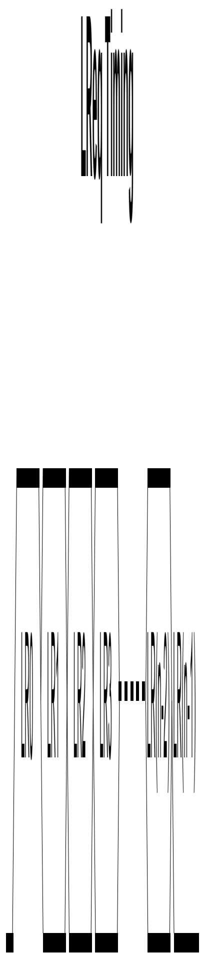

Figure I-2 – LReq timing

If the LReq transfer is a bus request, it is 7 bits long and has the following format:

Table I-4 – Bus request format

Bit(s)	Name	Description
0	Start Bit	Indicates start of transfer. Always 1.
1-3	Request Type	Indicates which type of bus request is being performed. See the table below for the encoding of this field.
4-5	Request Speed	The speed at which the phy will be sending the packet for this request. This field has the same encoding as the speed code from the first symbol of the receive packet. See the table below for the encoding of this field. This field can be expanded to support data rates higher than 400 Mbit/sec in the future.
6	Stop Bit	Indicates end of transfer. Always 0.

If the transfer is a read request, it is 9 bits long and has the following format:

Table I-5 – Read request format

Bit(s)	Name	Description
0	Start Bit	Indicates start of transfer. Always 1.
1-3	Request Type	Indicates that this is a register read See the table below for the encoding of this field.
4-7	Address	The internal phy address to be read.
8	Stop Bit	Indicates end of transfer. Always 0.

If the transfer is a write request, it is 17 bits long and has the following format:

Table I-6 – Write request format

Bit(s)	Name	Description
0	Start Bit	Indicates start of transfer. Always 1.
1-3	Request Type	Indicates that this is a register write See the table below for the encoding of this field.
4-7	Address	The internal phy address to be written.
8-15	Data	For a write transfer, the data to be written to the specified address.
16	Stop Bit	Indicates end of transfer. Always 0.

The request type field is encoded as follows:

Table I-7 – Request type field

LR[1:3]	Name	Meaning
000	TakeBus	Take control of the bus immediately upon detecting idle, Do not arbitrate
001	IsoReq	Arbitrate for the bus, no gaps
010	PriReq	Arbitrate after a fair gap, ignore fair protocol
011	FairReq	Arbitrate after a fair gap, following fair protocol
100	RdReg	Return specified register contents through status transfer
101	WrReg	Write to specified register
110-111	reserved	reserved

The request speed field is encoded as follows:

Table I-8 – Request speed field

LR[4:5]	Data Rate
00	100 Mbit/sec
01	200 Mbit/sec
10	400 Mbit/sec
11	>400 Mbit/sec

To request the bus for fair or priority access, the link sends the request at least one clock after the interface becomes idle. The link interprets the *receive* state on the Ctl pins as a lost request. If the link sees the *receive* state anytime during or after it sends the request transfer, it assumes the request is lost and reissues the request on the next *idle*. The phy will ignore a fair or priority request if it asserts the receive state anytime during the request transfer. Note that the minimum length of a packet is two clock cycles in the case of a 400 Mbit/secack packet. The minimum request packet is 8 clock cycles. It is important that the link and phy agree to interpret a lost request the same way.

The cycle master node uses a normal priority request to send the cycle start message. To request the bus to send isochronous data, the link can issue the request at any time after receiving the cycle start. The phy will clear an isochronous request only when the bus has been won.

To send an ack, the link must issue a TakeBus request during the reception of the packet addressed to it. This is required because the delay from end of packet to ack request adds directly to the minimum delay every phy must wait after every packet to allow an ack to occur. After the packet ends, the phy immediately takes control of the bus and grants the bus to the link. If the header CRC of the packet turns out to be bad, the link releases the bus immediately. The link cannot use this grant to send another type of packet. To ensure this, the link must wait 160ns after the end of the received packet to allow the phy to grant it the bus for the ack, then release the bus and proceed with another request.

Though highly unlikely, it is conceivable that two different nodes can perceive (one correctly, one mistakenly) that an incoming packet is intended for them and both issue an ack request before checking the CRC. Both nodes' phys would grab control of the bus immediately after the packet is complete. This condition will cause a temporary, localized collision of the data-on line states somewhere between two phys intending to acknowledge. All other phys on the bus would see the data-on state. This collision would appear as a 'ZZ' line state, and would not be interpreted as a bus reset. The mistaken node would drop it's request as soon as it has checked the CRC (maximum TBD µs) and the bogus line state would go away. The only side effect of such a collision would be the loss of the intended ack packet, which would be handled by the higher-layer protocol.

For write requests, the phy takes the value in the data field of the transfer and loads it into the addressed register as soon as the transfer is complete. For read requests, the phy returns the contents of the addressed register at the next opportunity through a status transfer. The link is allowed to perform a read or write operation at any time. If the status transfer is interrupted by an incoming packet, the phy continues to attempt the transfer of the requested register until it is successful.

Once the link issues a request for access to the bus (takebus, iso, fair, or priority) it cannot issue another request until the phy indicates "lost" (incoming packet) or "won" (transmit). The phy ignores new requests while a previous request is pending.

I.3.2 Status

When the phy has status information to transfer to the link, it will initiate a status transfer. The phy will wait until the interface is idle to perform the transfer. The phy initiates the transfer by asserting *status* (01b) on the Ctl pins, along with the first two bits of status information on D[0:1]. The phy maintains Ctl==*status* for the duration of the status transfer. The phy may prematurely end a status transfer by asserting something other than *status* on the Ctl pins. This should be done in the event that a packet arrives before the status transfer completes. There must be at least one *idle* cycle in between consecutive status transfers.

The phy normally sends just the first 4 bits of status to the link. These bits are status flags which are needed by link state machines. The phy sends an entire status packet to the link after a request transfer which contains a read request, or when the phy has pertinent information to send to the link or transaction layers. The only defined condition where the phy automatically sends a register to the link is after self-identification, where it sends the 'physicalID' register which contains the new node address.

The timing for the transfer is shown below.

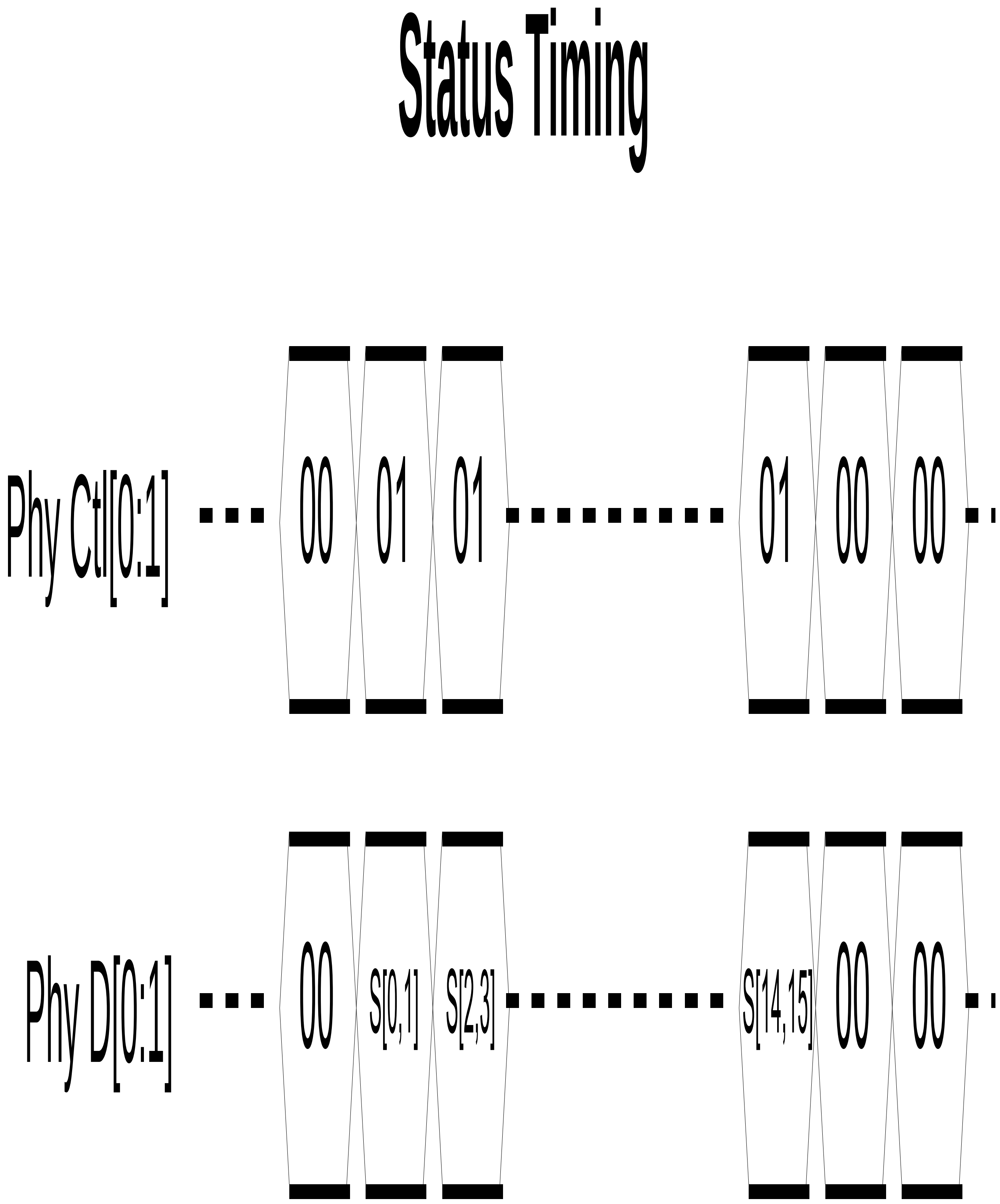

Figure I-3 – Status timing

Table I-9 – Status bits

Bit(s)	Name	Description
0	Arbitration Reset Gap	The phy has detected that the serial bus has been idle for an arbitration reset gap time. This bit is used by the link in the busy/retry state machine. The arbitration reset gap time is defined by the 1394 standard.
1	Fair Gap	The phy has detected that the serial bus has been idle for a fair gap time. This bit is used by the link to detect the end of an isochronous cycle. The fair gap time is defined by the 1394 standard.
2	Bus Reset	The phy has entered bus reset state
3	Phy Interrupt	The phy is requesting an interrupt to the host The interrupting conditions are TBD. Most likely used for "hung in tree-ID" for loop-like-thing detection.
4-7	Address	When transferring the contents of a register to the link, such as when responding to a read through the LReq pin, this field holds the address of the register being read.
8-15	Data	This field holds the data corresponding to the register being transferred.

I.3.3 Transmit

When the link requests access to the serial bus through the LReq pin, the phy arbitrates for access to the serial bus. If the phy wins the arbitration, it grants the bus to the link by asserting *transmit* on the Ctl pins for one SClk cycle, followed by *idle* for one cycle. After sampling the *transmit* state from the phy, the link takes over control the interface by asserting either *hold* or *transmit* on the Ctl pins. The link asserts *hold* to keep ownership of the bus while preparing data. The phy asserts the data-on state on the serial bus during this time. When it is ready to begin transmitting a packet, the link asserts *transmit* on the Ctl pins along with the first bits of the packet. After sending the last bits of the packet, the link asserts either *idle* or *hold* on the Ctl pins for one cycle then releases the interface. In the hold case, the link asserts idle for one cycle after asserting hold before tristating those pins. The *hold* state here indicates to the phy that the link needs to send another packet without releasing the bus. The phy responds to this *hold* state by waiting the required minimum time and then asserting *transmit* as before. This function would be used after sending an ack if the link intends to send a unified response, or to send consecutive isochronous packets during a single cycle. The only requirement when sending multiple packets during a single bus ownership is that all must be transmitted at the same speed, since the speed of the packet transmission is set before the first packet. When the link has finished sending the last packet for the current bus ownership, it releases the bus by asserting *idle* on the Ctl pins for a single SClk cycle. The phy begins asserting *idle* on the Ctl pins one clock after sampling *idle* from the link. Note that whenever the D and Ctl lines change 'ownership' between the phy and the link, there is an extra clock period allowed so that both sides of the interface can operate on registered versions of the interface signals, rather than having to respond to a Ctl state on the next cycle.

Note that it is not required that the link enter the *hold* state before sending the first packet if implementation permits the link to be ready to transmit as soon as bus ownership is granted. The timing for a single packet transmit operation is shown below. In the diagram, D0 through Dn are the data symbols of the packet, ZZ represents high impedance state.

Figure I-4 – Transmit timing

I.3.4 Receive

Whenever the phy sees the 'data-on' state on the serial bus, it initiates a receive operation by asserting *receive* on the Ctl pins and '1' on each of the D pins. The phy indicates the start of a packet by placing the speed code (encoding shown below) on the D pins, followed by the contents of the packet, holding the Ctl pins in *receive* until the last symbol of the packet has been transferred. The phy indicates the end of the packet by asserting *idle* on the ctl pins. Note that the speed code is a phy-link protocol and not included in the calculation of the CRC or other data protection mechanisms.

It is possible that a phy can see data-on appear and then disappear on the serial bus without seeing a packet. This is the case when a packet of a higher speed than the phy can receive is being transmitted. In this case, the phy will end the packet by asserting *idle* when the data-on state goes away.

If the phy is capable of a higher data rate than the link, the link detects the speed code as such and ignores the packet until it sees the *idle* state again.

The timing for the receive operation is shown below. In the diagram, SP refers to the speed code and D0 through Dn are the data symbols of the packet.

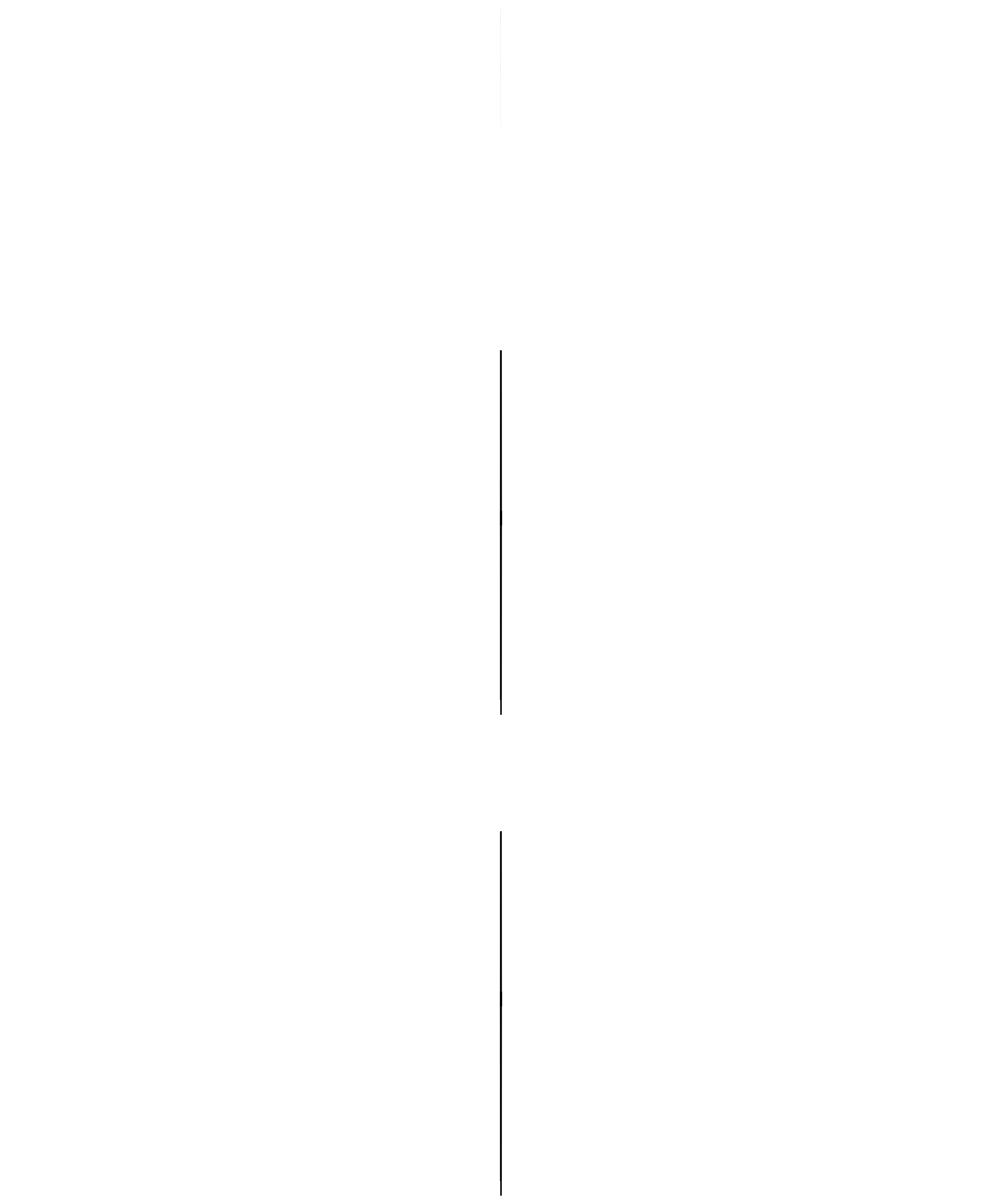

Figure I-5 – Receive timing

The speed code for the receive operation is defined as follows:

Table I-10 – Receive speed code

D[0:7]	Data Rate
00xxxxxx	100 Mbit/sec
0100xxxx	200 Mbit/sec
01010000	400 Mbit/sec
11111111	'data-on' indication

I.4 PhyRegister Map

The accessible registers in the phy are shown in the following diagram. All registers beyond the last port status (AstatN...) register are implementation dependent.

Figure I-6 – Phy register map

Table I-11 – Phy register fields

Field	Size	Type	Description
PhysicalID	6	r	The address of this node determined during self-identification
R	1	r	Indicates that this node is the root.
PS	1	r	Cable power status
RHB	1	rw	Root hold-off bit. Instructs the phy to attempt to become the root during the next bus reset
IBR	1	rw	Initiate bus reset. Instructs the phy to initiate a bus reset at the next opportunity
ATS	3	rw	Arbitration timer setting. Used to optimize gap times based on the size of the network. See 1394 standard for encoding of this field.
SPD	2	r	Indicates the top speed this phy can handle. Same encoding as for speed code in the request packet.
#Ports	4	r	The number of ports on this phy. This also indicates how port status registers will follow.
AStat<n>	2	r	TPA line state on port <n>
11 = ZZ	01 = 1	10 = 0	00 = invalid
BStat<n>	2	r	TPB line state on port <n> (same encoding as AStat<n>)
Ch<n>	1	r	If =1, port <n> is a child, else parent.
Con<n>	1	r	If =1, port <n> is connected, else disconnected.

I.5 State Diagrams

The following state diagrams describe the actions required of the interface in the link and the phy to support the functions described in this document.

I.5.1 Link Request

This diagram describes the how the link should behave for the various types of requests.

.

Figure I-7 – Link request state machine

Table I-12 – Link request state machine inputs

Name	Description
FairReq	Fair asynchronous request
PriReq	Priority asynchronous request
AckReq	Request to send an acknowledge
IsoReq	Isochronous data request
RWReq	Read or write register request
ReqSent	Entire request has been shifted out
Ctl	Interface control pins

Table I-13 – Link request state machine outputs

Name	Description
Won	Indicates bus ownership to transmitter
Lost	Indicates lost request to transmitter
ShiftEn	Enables LReq shift register

Table I-14 – Link request state machine state definitions

State	Name	Actions
R0	IDLE	none
R1	REQUEST	ShiftEn=true (send request stream to phy)
R2	LOST	Lost=true
R3	WAIT	none
R4	WON	Won=true, go to IDLE
R5	IMMREQBUSY	ShiftEn=true (send request stream to phy)
R6	IMMREQIDLE	ShiftEn=true (send request stream to phy)
R7	WAITBUSY	none
R8	WAITIMM	none
R9	RWREQ	ShiftEn=true (send read or write request to phy)

I.5.2 Link General

Figure I-8 – Link general state diagram

Table I-15 – Link general state machine inputs

Name	**Description**
DataReady	Transmitter can send on next cycle
PktDone	Transmitter is sending last bits of packet
MorePkts	Transmitter wants to send consecutive packets
Ctl	Interface control pins

Table I-16 – Link general state machine outputs

Name	**Description**
Tristate	Tristates D, Ctl pins
StatusShiftEn	Enables status shift register
DInShiftEn	Enables receive shift register
DOutShiftEn	Enables transmit shift register
Speed	Speed of packet to be received
Ctl	Interface control pins

Table I-17 – Link general state definitions

State	**Name**	**Actions**
L0	IDLE	tristate = true
L1	STATUS	tristate = true, statusShiftEn = true
L2	RECEIVE	tristate = true, dInShiftEn = true
L3	TRANSMIT	Ctl = transmit, DOutShiftEn = true
L4	HOLD	Ctl = hold
L5	RELEASE	Ctl = idle, go to IDLE
L6	CONTINUE	Ctl = hold, go to IDLE

I.5.3 Phy General

This state machine diagram describes how the phy handles the various events which happen in this interface. It purposely does not describe how the phy requests the bus or observes the required gaps.

Figure I-9 – Phy general state diagram

Table I-18 – Phy general state machine inputs

Name	**Description**
FairReq	Fair asynchronous request
PriReq	Priority asynchronous request
AckReq	Request to send an acknowledge
IsoReq	Isochronous data request
RdReq	Read register request
WrReq	Write register request
Receive	A packet is incoming
Grant	Arbitration won
ReadPending	A register read has been received
Ctl	Interface control pins

Table I-19 – Phy general state machine outputs

Name	**Description**
Tristate	Tristates D, Ctl pins
Request	Request bus on next idle (to rest of phy)
Ctl	Interface control pins

Table I-20 – Phy general state definitions

State	**Name**	**Actions**
P0	IDLE	Ctl = idle or receive
P1	REQUEST	Request = true, Ctl = idle or receive
P2	GRANT	Tristate = true
P3	HOLD	Ctl = transmit, DOutShiftEn = true
P4	WAIT	Ctl = receive
P5	STATUS	Send status transfer, clear ReadPending if successful

I.6 Isolation Barrier

.

Figure I-10 – Transformer Isolation Barrier Circuit Example

.

Figure I-11 – Capacitive Isolation Barrier Circuit Example

The above circuits demonstrate how to acheive galvanic isolation using two different methods. Each of these circuits provide isolation for one bidirectional connection. The logic on the enable to the output drivers acts as a digital differentiator to drive the output only when it changes. The differentiator is disabled by the 'Direct' signal when the chips are directly connected. The first of the two circuits above uses transformers to isolate the phy from the link. The second uses capacitors, and would be more cost-effective than the transformer version. The transformer-based version is designed to handle 500v of ground potential difference between the link and phy. The capacitor-based version can tolerate 60v.

I.7 AC Timing

The protocol of this interface is designed such that all inputs and outputs at this interface can be registered immediately before or after the I/O pad and buffer. No state transitions need be made that depend directly on the chip inputs and chip outputs can come directly from registers without combinational delay or additional loading. This configuration provides generous margins on setup and hold time. The timing defined here assumes there some delay from the SClk input to the input registers due to a clock tree. As a result, the timing below provides for little or no setup time and generous hold time. The SClk output from the phy should originate directly from the root of the phy's internal clock tree to provide similar insertion delay on inputs to the phy.

The AC timing parameters for this interface also assume a maximum delay through an isolation barrier.

Table I-21 – AC timing

Parameter	Unit	Min	Max
D, Ctl, LReq setup to SClk rise	ns	0	
D, Ctl, LReq hold to SClk rise	ns	10	
SClk rise to D, Ctl, LReq out	ns		10
Delay through isolation barrier	ns		3
Hysteresis input rising threshold	v	Vcc/2 + 0.2	Vcc/2 + 1.1
Hysteresis input falling threshold	v	Vcc/2 - 1.1	Vcc/2 - 0.2

I.8 Open Issues

State machines should be revisited.

I.9 Revision History

1.5.2	16 Feb 93	Creation of revision history. Added data-on state to receive operation.
	23 Mar 93	Added 'Direct' signal to disable differentiator.
1.5.3	14 Apr 93	Made LReq active high to prevent false request when link goes down. Fiddled with phy address map to incorporate speed, root bit, and #ports.
1.5.4	19 Apr 93	removed the fake speed code from the status transfer. (Wasn't taking advantage of it anyway, and the desire for fast status transfer won out.
1.5.5	10 May 93	Corrected isolation barrier diagram to demonstrate the desired implementation
1.5.6	18 May 93	Added "idle" state when transferring control, modified differentiator schematic to match.