IEEE 1394
Link Core
Specification

johnsonw10@opencores.org

Revision History

Rev	Date	Author	Description
0.1	11/11/01	Jim	First draft

Tabel Of Contents

1 Introduction

1.1 Requirements

1.1.1 Physical Interface

The Physical Interface (PHY) must meet the following requirements:

- Supports 100, 200, and 400 Mbits/s transfer
- IEEE 1394 compliant interface timing

1.1.2 Link Control

The Link Control must meet the following requirements:

- Supports all IEEE 1394 packets
- Generates and checks 32-bit CRC
- Supports asynchronous and isochronous transfers
- Acts as IEEE 1394 cycle master
- Detects lost cycle start message
- Provides programmable sizes for transmit and receive FIFOs

1.1.3 Design For Test

The core must have the following design-for-test (DFT) features.

- TBD

1.2 Reference

- 1394 Trade Association website (http://www.1394ta.org/)
- Agere Systems 1394 website (http://www.agere.com/1394/index.html)
- Texas Instruments 1394 website (http://www.ti.com/sc/1394)

1.3 Acronyms

CRC	Cyclic Redundancy Check
DFT	Design For Test
PHY	Physical Interface

2 Architecture

2.1 Block Diagram

The link layer is partitioned to two cores, a Link Core and a Host Controller Core. The Link Core is responsible for the communications and operations between the link layer and the physical layer. The Host Controller is responsible for the communications and services between the link layer and a host controller. Only the link core is covered in this specification.

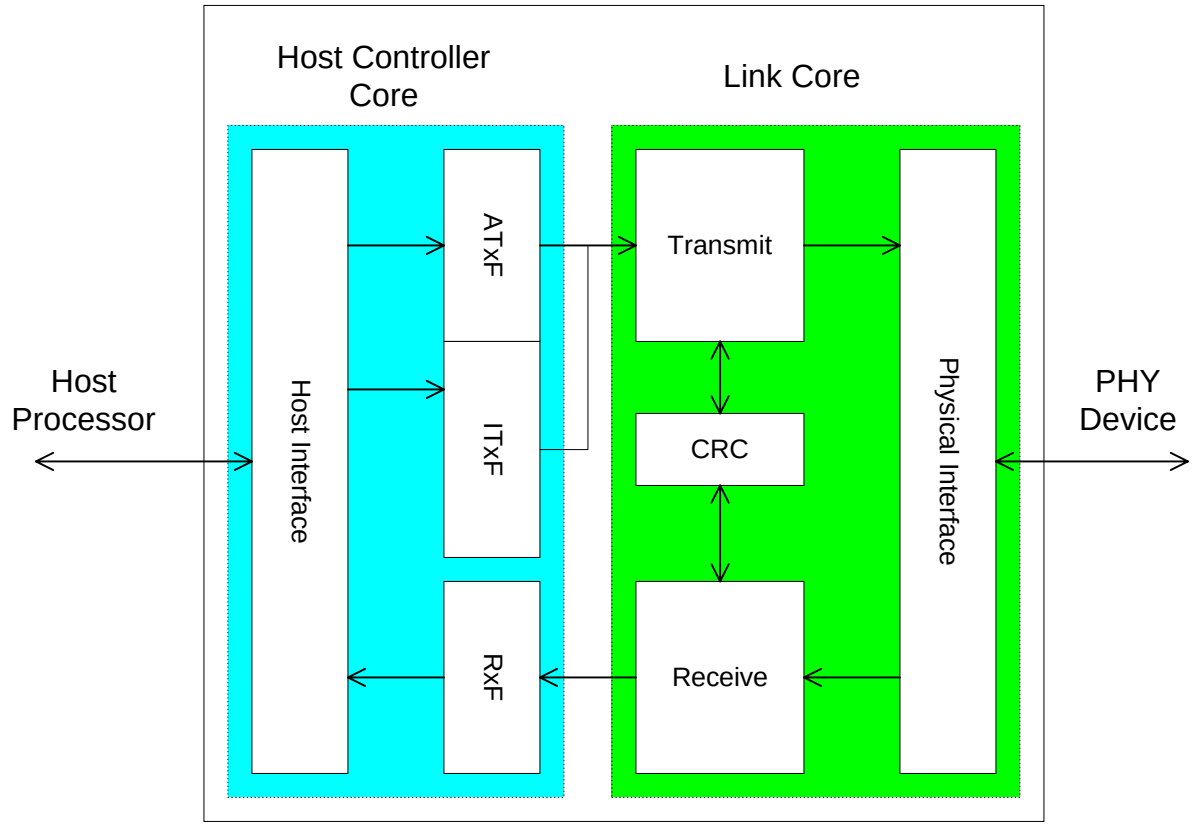


Figure 2.1 Link Layer Cores Block Diagram

2.2 Physical Interface

The physical (PHY) interface is responsible for the following operations:

- Gain access to the serial bus.
- Send and receive packet.
- Send and receive acknowledge packets.
- Provide read and write access to PHY registers.

2.3 Transmit

The transmit block is responsible for the following operations:

- Receive data from either the Asynchronous Transmit FIFO (ATxFIFO) or the Isochronous Transmit FIFO (ITxFIFO).
- Creates serial-bus packets to be transmitted through the PHY interface.
- Arbitrate for the serial bus to correctly send both asynchronous and isochronous packets
- Sends cycle-start packets in cycle master mode.

2.4 Receive

The receive control block is responsible for the following operations:

- Receive incoming packets from the PHY interface.
- Check the CRC of the incoming packets addressed to the node.
- Sends the header to Receiving FIFO (RxFIFO) if header CRC is good. Otherwise flush the header and ignore the rest of the packet.
- Check the rest data of the packet and sends the data to RxFIFO
- Send a status quadlet to RxFIFO.

2.5 CRC

The CRC block is responsible for a 32-bit CRC generation and checking for error detection. It generates the header and data CRC for transmitting packets and checks the header and data CRC for received packets.

3 I/O Ports

3.1 Physical Interface

Name	I/O	Description
D[0:7]	I/O	PHY-link interface data bus. Data is expected on D[0:1] for 100 Mbit/s packets, D[0:3] for 200 Mbit/s, and D[0:7] for 400 Mbits/s.
Ctl[0:1]	I/O	PHY-link interface control bus. Ctl[0:1] indicate the four operations that can occur on this interface.
LReq	O	Link request to PHY. LReq makes bus requests and register access requests to the PHY.
SClk	I	System clock. SClk is a 49.152-MHz clock from the PHY and used to generate the 24.576-MHz clock.

Table 3.1 PHY ports description

3.2 Host Interface

4 Link/PHY Interface Operations

All control and data signals are synchronized to and sampled on the rising edge of SClk.

The Ctl[0:1] control the flow of information and data between the Link Layer and Physical Layer. The encoding of Ctl[0:1] is shown in tables 4.xxx and 4.xxx.

Ctl[0:1]	Name	Description
00	Idle	No activity.
01	Status	Status information is being sent from the PHY to the link.
10	Receive	An incoming packet is being sent from the PHY to the link.
11	Transmit	The link is granted the control of the bus to send an outgoing packet.

Table 4.1 Ctl[0:1] encoding when PHY has control

Ctl[0:1]	Name	Description
00	Idle	Transmission is completed. The link releases the bus.
01	Hold	The link is holding the bus while data is being prepared for transmission, or indicating that another packet is to be transmitted without arbitrating.
10	Transmit	The link is sending an outgoing packet the PHY.
11	Unused	Unused

Table 4.2 Ctl[0:1] encoding when link has control

The D[0:7] is used to transfer status information, control information, or packet data between the devices. For 100 Mbit/s packets, D[0:1] are used data transfer; For 200 Mbit/s packet, D[0:3] are used data transfer; For 400 Mbit/s packets, D[0:7] are used for data transfer.

The LReq port sends serial service requests to the PHY for packet transmission, or PHY register read and write. The LReq port is normally low.

4.1 Request

To request access to the bus or to access a PHY register, the link sends a serial bit stream to the PHY on the LReq port (LR0, LR1, …, LRn-2, LRn-1). Bit 0 is the most significant and is transmitted first in the request bit stream. The length of the bit stream varies depending on the type of request: bus request, read request, or write request.

Regardless of the type of request, A start-bit of 1 is required at the beginning of the stream, and a stop-bit of 0 is required at the end of the stream. The second through fourth bits of the request stream indicate the type of the request. The request type field encoding is shown in table 4.xxx.

LR[1:3]	Name	Description
000	ImmReq	Immediate bus request. Upon detection of idle, the PHY

		takes control of the bus immediately without arbitration. Used for acknowledge transfers.
001	IsoReq	Isochronous bus request. Upon detection of idle, the PHY arbitrates for the bus without waiting for a subaction gap. Used for isochronous transfers.
010	PriReq	Priority bus request. The PHY arbitrates for the bus after a subaction gap, ignores the fair protocol. Used for cycle master request.
011	FairReq	Fair bus request. The PHY arbitrates for the bus after a subaction gap, follows the fair protocol. Used for fair transfers.
100	RdReq	The PHY returns the specified register contents through a status transfer.
101	WrReq	Write to the specified register.
110-111	Reserved	Reserved

Table 4.3 Request type field encoding

4.1.1 Bus Request

The bus request is a 7 bits long transfer as shown in table 4.xxx.

Bit	Name	Description
0	Start Bit	Start of transfer. Always 1.
1-3	Request Type	Type of bus request. See table 4.xxx
4-5	Request Speed	The speed at which the PHY will be sending the data for this request. See table 4.xxx.
6	Stop Bit	End of transfer. Always 0.

Table 4.4 Bus request format

The request speed encoding is shown in table 4.xxx.

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

Table 4.5 Request speed encoding

4.1.2 Read Request

The read request is a 9 bits long transfer as shown in table 4.xxx.

Bit	Name	Description
0	Start Bit	Start of transfer. Always 1.
1-3	Request Type	Type of bus request. See table 4.xxx
4-7	Address	The PHY register address to be read

8	Stop Bit	End of transfer. Always 0.

Table 4.6 Read request format

4.1.3 Write Request

The write request is a 17 bits long transfer as shown in table 4.xxx.

Bit	Name	Description
0	Start Bit	Start of transfer. Always 1.
1-3	Request Type	Type of bus request. See table 4.xxx
4-7	Address	The PHY register address to be read
8-15	Data	The data to be written to the specified address for a write transfer.
16	Stop Bit	End of transfer. Always 0.

Table 4.7 Bus request format

4.2 Status

When the PHY has status information to transfer to the link, it initiates a status transfer. The PHY waits until the interface is idle before starting the transfer. The PHY initiates the transfer by asserting *status* (01b) on Ctl, 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 Ctl. This occurs if a packet is received before the status transfer completes. The PHY continues to attempt to complete the transfer until all status information has been successfully transmitted. There is at least one idle cycle between consecutive status transfers.

The PHY normally sends just the first four bits of status to the link. These bits are status flags that are needed by link state machines. The PHY sends an entire 16-bit status packet to the link after a read register 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-ID, where the PHY sends the “physical-ID” register that contains the new node address.

The definitions of bits in the status transfer are shown in table 4.xxx:

Bit	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.
1	Subaction Gap	The PHY has detected that the serial bus has been idle for a subaction gap time. This bit is used by the link to detect the end of an isochronous cycle.
2	Bus Reset	The PHY has entered bus reset start state.
3	State Time-out	The PHY stayed in a particular state for too long a

		period. (e.g. a loop in the cable topology)
4-7	Address	This field holds the address of the PHY register being transferred.
8-15	Data	This field holds the data of the PHY register being transferred.

Table 4.8 Status bits definitions

4.3 Transmit

When the link requests access to the serial bus through LReq, the PHY arbitrates for access to the bus. If the PHY wins arbitration, it grants the bus to the link by asserting the *transmit* state (11b) on Ctl for one SClk cycle, followed by *idle* (00b) for one clock cycle. The PHY then takes over control of the bus by asserting either *hold* (01b) or *transmit* (10b) on Ctl. The link asserts *hold* to retain control of the bus while preparing data for transmission. The PHY asserts data-on state on the bus during this time. When the link is ready to send data, it asserts *transmit* on Ctl and sends the first bits of packet data. The *transmit* state is held on Ctl until the last bits of data has been sent. The link then asserts either *hold* or *idle* on Ctl for one clock cycle, and then asserts *idle* for one additional cycle before tristating Ctl and D ports

The *hold* state asserted at the end of packet transmission indicates to the PHY that the link needs to send another packet without releasing the serial bus. The PHY responds to the request by waiting the required minimum packet separation time and then asserting *transmit* as before. This function may be used to send a unified response after sending an acknowledge, or to send consecutive isochronous packets during a single cycle. All packets transmitted during a single bus ownership must be of the same speed, since the speed of the packet 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 Ctl for two SClk cycles. The PHY begins asserting *idle* on the Ctl one clock cycle after sampling *idle* from the link. Note that whenever the D and Ctl change direction 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.

4.4 Receive

Whenever the PHY detects the data-on state on the serial bus, it initiates a receive operation by asserting *receive* on Ctl terminals and a “1” on the D ports (data-on indication). The PHY indicates the start of a packet by placing the speed code (see table 4.xxx) on the D ports, followed by the packet data. The PHY holds Ctl in the *receive* state until the last symbol of the packet has been transferred. The PHY indicates the end of packet data by asserting *idle* on Ctl. Note that the speed code is part of the PHY-Link protocol and is not included in the calculation of CRC or any 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 the packet speed exceeds the capability of

the receiving PHY. In this case, the PHY will end the packet by asserting idle when the data-on state goes away.

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

Table 4.9 Receive speed encode

5 Link/Host Packet Formats

5.1 Packet Components

5.1.1 Transaction Label (tl)

The transaction label specifies a unique tag for each outstanding transaction.

5.1.2 Retry Code (rt)

The retry code specifies whether this packet is a retry attempt and the retry protocol to be followed by the destination node.

5.1.3 Transaction Code (tcode)

The transaction code specifies the packet format and the type of transaction.

5.1.4 Priority (pri)

The priority only has meaning for the backplane PHY.

5.1.5 Destination ID (destination_ID)

The destination ID specifies the node ID of the receiving node. It is the concatenation of the destination bus_ID and destination physical_ID.

5.1.6 Destination Offset (destination_offset)

The destination offset specifies the lower 48 bits of the destination node address of a request packet.

5.1.7 Response Code (rcode)

The response code specifies the response to an earlier corresponding request.

5.2 Asynchronous Transmit Packets

5.2.1 Asynchronous Packets with No-data Payload

5.2.1.1 Read Request for Data Quadlet

0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
Reserved				tl	rt	tcode	pri
destination_ID				destination_offset_high			
destination_offset_low							

Table 5.1 Quadlet Read Request Packet

5.2.1.2 Write Response

0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
Reserved				tl	rt	tcode	pri
destination_ID				rcode	Reserved		
Reserved							

Table 5.2 Quadlet Write Response Packet

5.2.2 Asynchronous Packets with Data Quadlet Payload

5.2.2.1 Read Request for Data Block

0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
Reserved				tl	rt	tcode	pri
destination_ID				destination_offset_high			
destination_offset_low							
data_length				extended_tcode			

Table 5.3 Read request for data block Packet

5.2.2.2 Write Request for Data Quadlet

0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
Reserved				Tl	rt	tcode	pri
destination_ID				destination_offset_high			
destination_offset_low							
quadlet_data							

Table 5.4 Write request for data quadlet packet

5.2.2.3 Read Response for Data Quadlet

0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
Reserved				Tl	rt	tcode	pri
destination_ID				rcode	Reserved		
Reserved							
quadlet_data							

Table 5.5 Read response for data quadlet packet

5.2.3 Asynchronous Packets with Data Quadlet Payload

5.2.3.1 Write Request for Data Block

0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
Reserved				tl	Rt	tcode	pri
Destination_ID				destination_offset_high			
destination_offset_low							
data_length				extended_tcode			
data field							
			zero pad bytes (if necessary)				

Table 5.6 Write request for data block packet

5.2.3.2 Read Response for Data Block

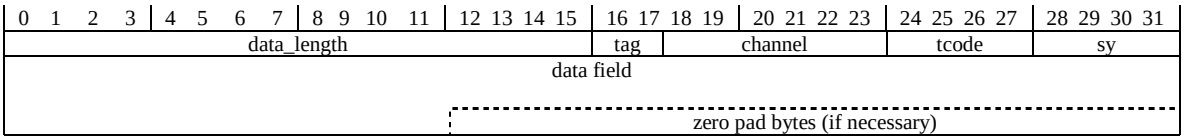
0 1 2 3	4 5 6 7	8 9 10 11	12 13 14 15	16 17 18 19	20 21 22 23	24 25 26 27	28 29 30 31
Reserved				tl	rt	tcode	pri
destination_ID				rcode	Reserved		
Reserved							
data_length				extended_tcode			
data field							
			zero pad bytes (if necessary)				

Table 5.7 Block Read Request Packet

5.3 Asynchronous Receive Packets

5.4 Isochronous Packets

5.4.1 Isochronous Data Block Packet



6 Register Map

Name	Offset	Width	Access	Description
node_ID	0x4	32	R/W	

6.1 node_ID Register

Bits	Name	Description
[0:9]	bus_ID	The bus_ID is used with the node_ID as the source_ID for outgoing packets and to accept or reject incoming packets. The bus_ID 0x3FF indicates the local bus.
[10:15]	node_ID	The node_ID is used with the bus_ID as the source_ID for outgoing packets and to accept or reject incoming packets. Packets with node_ID 0x3F are broadcast packets. The node_ID is automatically set to the node's physical_ID by a PHY register 0 transfer after bus reset.
[16:31]	Reserved	Reserved. Set to 0.

7 Implementation