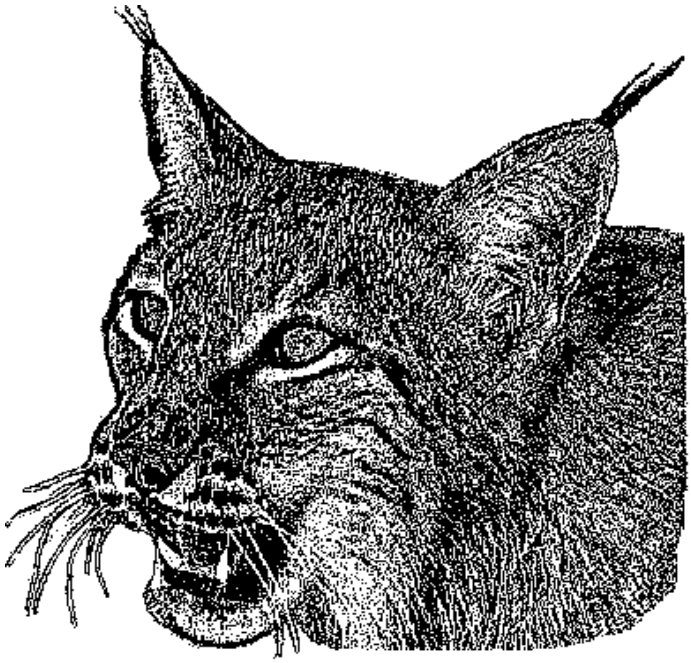
TEXAS INSTRUMENTS



PCI-LYNX ASIC
F643950
Functional Specification

Revision 0.10

March 28, 1996

Table Of Contents

List Of Figures

1. Revision History

Revision 0.02			
Section	Date	Editor	Changes Summary
Appendix E, 6.4, 6.1	02/07/95	Henry Angulo	Changed FIFO register definitions to allow slave read or write access in normal operating mode. Deleted test register bits which were used to enable FIFO read-write slave accesses. Added master error status bit to the ASYNC and ISOCH transmit control token formats.

Revision 0.03			
Section	Date	Editor	Changes Summary
Appendix D,E 5.2.2.1, 6.3, 6.5.6	02/20/95	Henry Angulo	Deleted SNOOP packet format and associated FIFO control token definitions Updated DMA section after state diagrams with text file received from Randy Pipho and updated DMA control register definitions. Removed Link interrupts from global interrupt status and Enable registers and replaced with single Link interrupt status and enable. Added Link enable and Status registers to Link register definitions.

Revision 0.04			
Section	Date	Editor	Changes Summary
6.1, 6.5, 5.2.4.1, 5.2.4.2 5.2.4.3 APPENDIX C APPENDIX B 5.2.1.6 , 6.2.15 5.2	4/3/95	Henry Angulo	Added control registers for implementing busy-retry re-transmit control Update PCL map in DMA section to show Alternate next PCL to branch to on async transmit retry timeouts. Added high level functional block diagram of DMA channel Receive Packet comparator function. Changed Package pictorial from 144pin to 176 pin QFP Updated Aux port signals to match the current definition for this port. Changed pin-assignments to reflect 176 pin package. Also Added a signal description list. Update Functional Partition block diagram to show aux port and added aux port verbal description hardware function description Added Aux port Control and status register bit definitions
6.4 APPENDIX E 5.2.2 figures 4 and 5	4/3/95	Henry Angulo	Re-worked FIFO control and status register definitions to simplify the PCI slave interface required to control and monitor the FIFO functionality. Added A table of ack codes returned to transmit DMA channel for ASYNC transmit packets Added GPIO ready condition to PCL control and byte count definition

Revision 0.05			
Section	Date	Editor	Changes Summary
5.0 6.0 and Appendix E	5/4/95	Henry Angulo	Changed FIFO architecture from 4 to 3 FIFO's. This also impacted the DMA controller to some extent

Revision 0.06			
Section	Date	Editor	Changes Summary
5.0 and 6.0	7/6/95	Chuck Storvik	Updated AUX port, added ZV port, misc changes to PCI

Revision 0.07			
Section	Date	Editor	Changes Summary
5.0	7/10/95	Chuck Storvik	Added GPIO
	8/7/95	Richard Baker	Added Autoboot section, PCL coding examples appendix, misc register updates
	8/10/95	Randy Pipho	Modified PCL format to accommodate returned remaining transfer count and next buffer address entries.
	8/17/95	Henry Angulo	Re-worked link-layer and FIFO control-status register definitions.

Revision 0.08			
Section	Date	Editor	Changes Summary
6.0, Appendix E	10/13/95	Henry Angulo	Updated 1394 Link layer control-status register definitions and fifo control token formats. Merged Busy Retry re-XMT Count and Interval Delay regs into 1 register @ F20.
*	10/24/95	Chuck Storvik	Reworked AUX/Local Bus Control register/GPIO bits.
*	10/25/95	Mark Young	Corrected errors and reversed many tables to list Bit 31 on top instead of Bit 00. Removed 10us timer. Renamed LINK_INT bit in PCI and 1394 Interrupt registers. Added Interrupt Logic section. Changed all fixed APPENDIX references to “inserted”.
*	1/11/96	Richard Baker	Added appendix on Serial EEPROM address map
6.4, 6.5	1/18/96	Henry Angulo	Implemented changes to FIFO and Link layer control register definition tables.

Revision 0.09			
Section	Date	Editor	Changes Summary
App B	01/26/96	Burke Henehan	Updated pinout, etc

Revision 0.10			
Section	Date	Editor	Changes Summary
Title Sheet	03/28/96	Burke Henehan	Corrected symbolized part number to F643950

2. Introduction

2.1 Scope Of Document

This document, upon acceptance and release, shall specify the requirements for the design of an Application Specific Integrated Circuit (ASIC) to be implemented using Texas Instruments Inc TGC3000T ASIC technology. This device shall perform the primary function of controlling the transfer of 1394 data packets between devices operating in a PCI bus environment and devices operating in a ieee 1394 bus environment. The device here after shall be referred to as the PCI-LYNX.

2.2 Feature Set

The PCI-LYNX device shall support the following features:

- Compliant with IEEE 1394-1995
- Compliant with PCI specification revision 2.0
- Performs function of 1394 cycle master
- Detect lost cycle start messages
- Generates 32 bit CRC for transmission of 1394 packets
- Performs 32 bit CRC checking on reception of 1394 packets
- Supports isolation barrier between the PCI-LYNX and the PHY layer
- Supports ieee 1394 transfer rates of 100, 200 and 400 mbps
- Provides three size programmable FIFO's (ASYNC transmit + ISO transmit + General Receive)
- Programmable 5 channel address comparator logic for receiving incoming 1394 packets and assigning them to a DMA channel.
- Provides 5 scatter-gather DMA channels where the 1394 operation of each channel can be programmed to support:
  - Asynchronous Packet transmit
  - Isochronous Packet Transmit
  - Asynchronous Packet Receive
  - Isochronous Packet Receive
- Provides PCI bus master function for supporting DMA operations
- Provides PCI slave function for read-write access of internal registers
- Implements a 32-bit PCI address-data path.
- Provides PCI address-data parity checking
- Provides Software control of interrupt events
- Supports Plug and Play specification
- Provides a programmable external Local Bus for implementing a dedicated data path to external logic (i.e. SRAM, ROM, etc.)
- Provides an 8 / 16-bit Zoom Video (ZV) Port for the transferring of video data directly to an external motion video memory area
- Operate from 3.3 Volt power while maintaining 5 Volt tolerant inputs

2.3 Applicable Documents

The following documents shall apply to the design of the PCI-LYNX ASIC.

- PCI Bus specification revision 2.0
- IEEE STD 1394 - 1995 High Performance Serial Bus

- IEEE STD 1212-1991, IEEE Standard Control and Status Register (CSR) Architecture for Microcomputer Busses
- Texas Instruments Inc. TGC3000 ASIC Design Manual

3. Performance Requirements

The PCI-LYNX device shall meet the following performance requirements

- Maximum PCI Bus Acquisition Latency Allowed - TBD
- Maximum DMA read PCI Bus Transfer Rate - TBD
- Maximum DMA write PCI Bus Transfer Rate - TBD
- 1394 Serial Data transfer Rates - 100 mbps 200 mbps
- Phy Link Interface Clock Frequency - 50 MHz
- PCI Interface Clock rate - 0 to 33 MHz

4. Mechanical Requirements

4.1 Packaging Requirements

The PCI-LYNX ASIC shall be implemented as 176-pin plastic quad flat pack (PQFP) package. The outline dimensions for this package are provided in APPENDIX C - ASIC Package Outline Dimension Drawing on page 106.

4.2 Pin Assignment Requirements

The PCI-LYNX ASIC shall implement the signal-to-pin assignments as shown in APPENDIX B - SIGNAL TO PACKAGE PIN ASSIGNMENTS on page 103.

5. Hardware Functional Description

5.1 System Overview

The following diagram provides a system overview of the PCI-LYNX ASIC as it would appear in a typical system configuration.

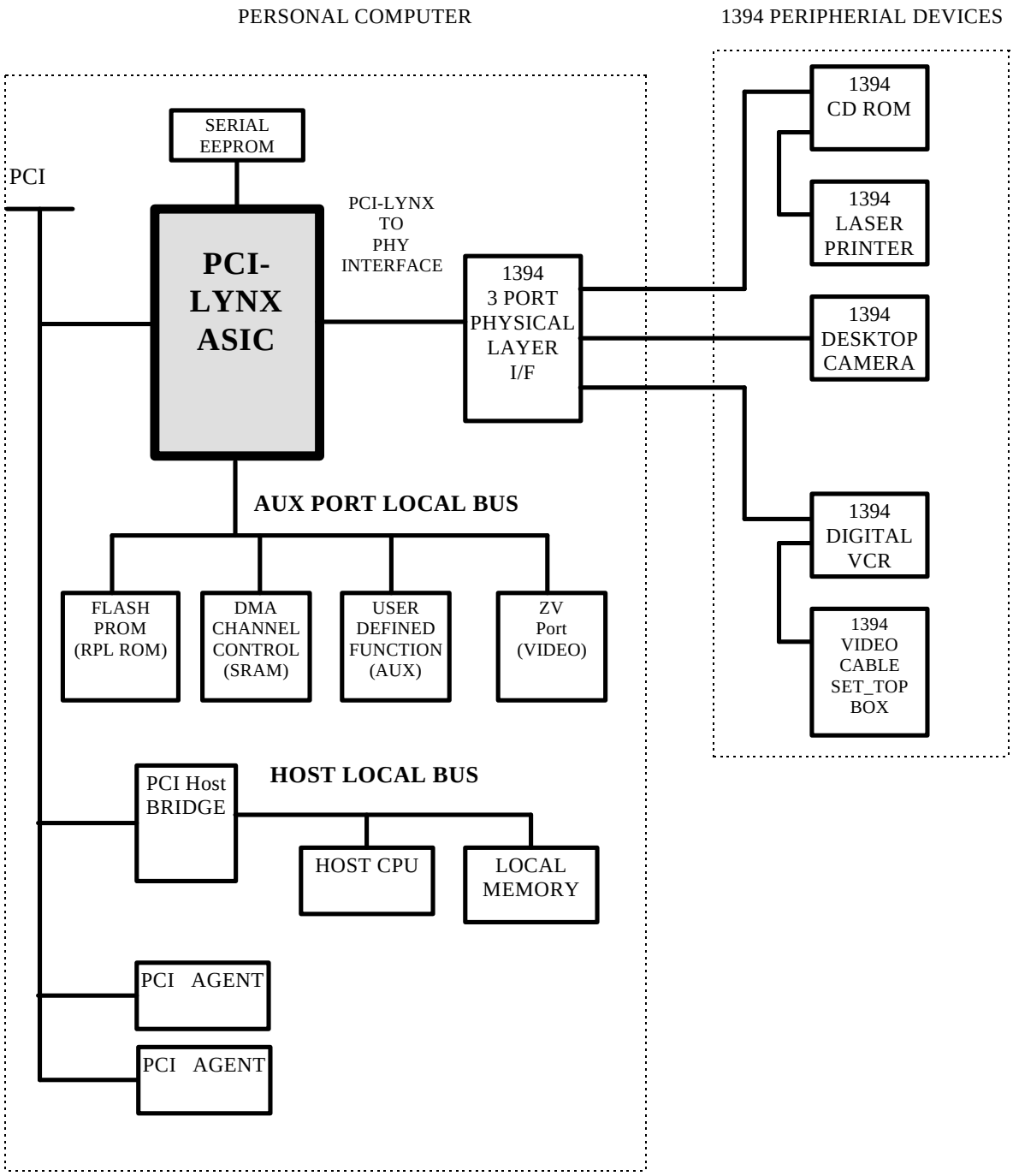
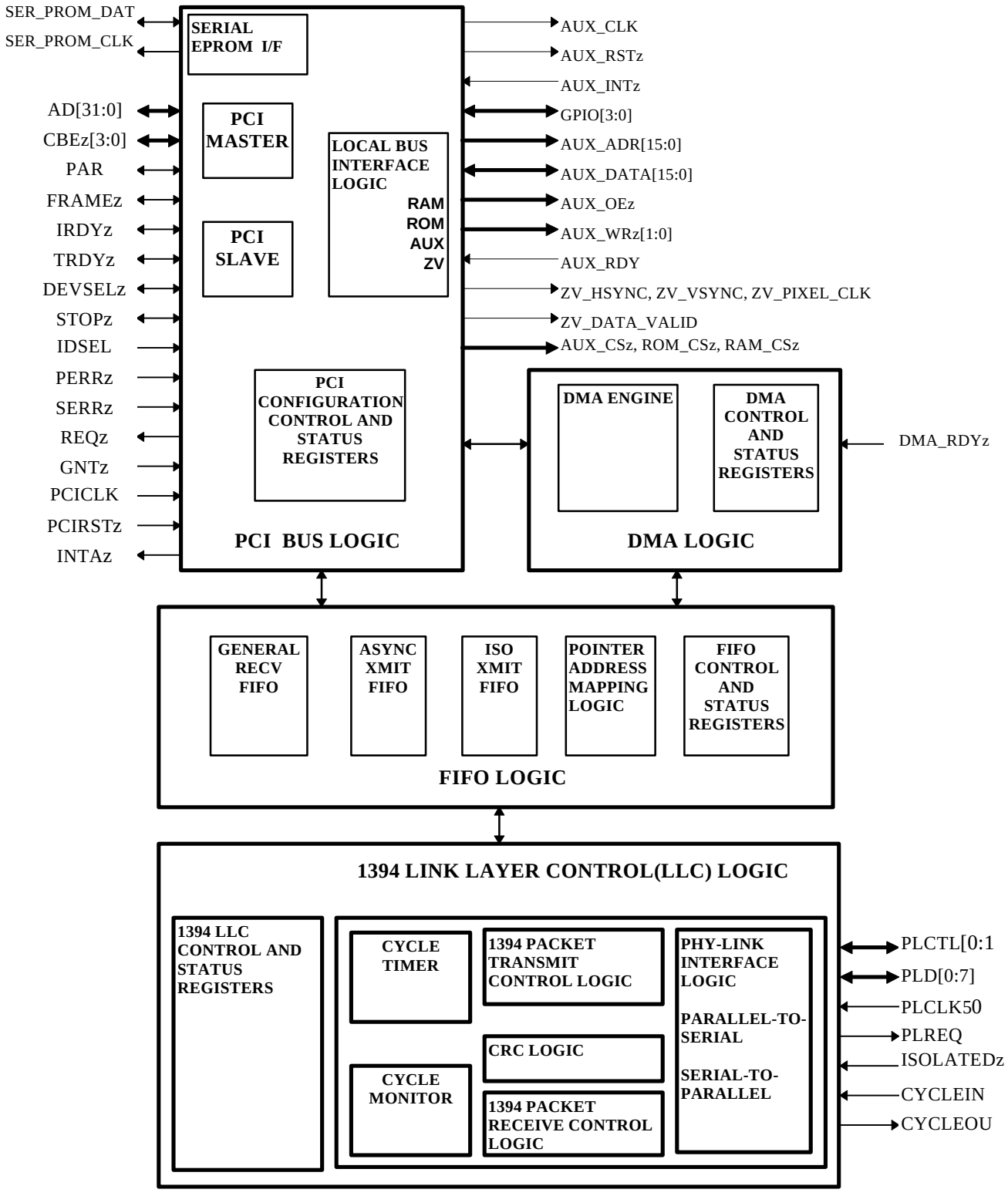


Figure 1. PCI-LYNX ASIC In A Typical System Configuration

5.2 ASIC FUNCTIONAL PARTITIONING

The following is a block diagram that shows the functional partitioning of the PCI-LYNX ASIC.

Figure 2: PCI-LYNX FUNCTIONAL PARTIONING



5.2.1 PCI Bus Logic

This functional block shall implement the logic required to interface the PCI-LYNX ASIC to the PCI bus. The PCI bus logic shall be designed to meet the requirements of PCI specification rev2.0. The functional partitioning of the PCI bus logic shall be as follows:

- Read and write slave interface control logic for accessing all of the PCI-LYNX control and status registers which are required by application software to control the operation of the PCI-LYNX and monitor it operational status.
- Bus master logic to provide the DMA logic with capability to initiate data transfers over the PCI bus as a master device.
- PCI configuration registers for use by system and application software for configuring and programming the PCI-LYNX. This includes the PCI required control and base registers as well as PCI-LYNX interrupt control and status and miscellaneous control and status registers.
- Auxiliary port to interface and control the RAM, ROM, AUX, ZV port, and GPIO interfaces.
- Serial EEPROM interface for power-up PCI configuration data and constant system control register information.

5.2.1.1 PCI Master Logic

This logic function shall implement the control logic required for the PCI-LYNX to operate on the PCI bus as a master device. This logic function shall meet the functional requirements for a PCI bus master device as specified in PCI specification rev2.0. As bus master, the following PCI Bus commands shall be supported:

PCI Bus Operation	CMD[3:0]	PCI-LYNX MASTER FUNCTIONS
Memory read	0110	DMA read from memory
Memory write	0111	DMA write to memory
Memory read line	1110	DMA read from memory
Memory write line and invalidate	1111	DMA write to memory

5.2.1.2 PCI Slave Logic

This function shall implement the control logic required for the PCI-LYNX device to operate on the PCI bus as a slave device. The logic for this function shall meet the functional requirements for a PCI slave device as specified in PCI specification rev 2.0. The PCI-LYNX as a slave device shall not decode the I/O read and write commands. The following commands shall be supported:

PCI Bus Operation	CMD[3:0]	PCI-LYNX SLAVE FUNCTIONS
Memory Read	0110	Memory read of PCI-LYNX addressed resource
Memory Write	0111	Memory write to PCI-LYNX addressed resource
Configuration Read	1010	Configuration read of PCI-LYNX addressed resource
Configuration Write	1011	Configuration write to PCI-LYNX addressed resource
Memory Read Multiple	1100	Memory read multiple to PCI-LYNX addressed resource
Memory Read Line	1110	Memory read line to PCI-LYNX addressed resource
Memory write line and invalidate	1111	Memory write to PCI-LYNX addressed resource

The PCI slave logic shall perform burst slave transfers when enabled by the ENA_SLV_BURST bit in the miscellaneous control register (at offset 0x40). The PCI slave logic shall perform posted write operations when possible when enabled by the ENA_POST_WR bit in the miscellaneous control register.

5.2.1.3 PCI Configuration Control and Status Registers

The LYNX PCI configuration register set is defined starting on page 64. These registers shall provide system and application software with the capability to program the PCI operational configuration of the PCI-LYNX. The functional behavior of these registers shall conform to the requirements of the PCI specification.

5.2.1.4 Serial EEPROM Interface

The serial EEPROM interface provides communication between the ASIC and an attached serial EEPROM. The serial EEPROM resides on an industry standard 2 wire serial bus at slave address 0.

At power-up, the serial EEPROM interface initializes a small number of locations in the PCI configuration registers from the EEPROM. While the serial EEPROM state machine is accessing the EEPROM, any incoming PCI slave access is terminated with retry status. A software reset will also initiate a reload of the PCI configuration register values from the serial EEPROM.

PCI Configuration Registers/fields initialized From the serial EEPROM:

1. PCI Subsystem ID
2. PCI Subsystem Vendor ID
3. PCI Maximum Latency
4. PCI Minimum Grant
5. ROM Control

This serial EEPROM also contains configuration data required by the 1394 Command Status Registers as specified in APPENDIX G - SERIAL EEPROM ADDRESS MAP on page 120 and optional manufacturing data. This information is read and written by the host processor emulating the 2 wire serial bus protocol through the serial EEPROM control register. The 2 wire serial bus is manipulated from the host processor by setting the serial EEPROM output enable bit to a "1", and then accessing the DATA and CLOCK bits to emulate the 2 wire serial bus protocol. Please reference "Phillips I^2C Peripherals Manual".

The 5us timer bit in the control register provide a timing reference for timing the 2 wire serial bus protocol events, if a more accurate source is not available. Since this timer is based on the PCI clock, it may be longer than desired, depending on the frequency of the PCI clock. After being written to 0, the timer bit will be set to 1 after the appropriate time delay. Host software may poll this bit to determine when the required time has passed for implementing the 2 wire serial bus protocol.

5.2.1.5 Local Bus Interface Logic

The PCI-LYNX Local Bus interface logic is a group of special I/O ports that share common logic. These ports are accessible from either the PCI bus or the DMA engine; these ports cannot function as master devices. These ports shall allow the PCI-LYNX to be connected to external devices or interfaces to provide for autonomous data transfers to/from such devices.

All local bus interfaces, except the Zoom Video Port (ZV Port), are synchronous to the LOCAL_CLK (a buffered version of PCI clock). The ZV Port clock is programmed to be based on versions of the PCI clock, 1394 clock, or an external clock.

The local bus provides the following I/O ports:

- **RPL ROM**
  - A 16 bit address bus and an 8 or 16 bit read or write data bus
  - Byte addressable/writable
  - programmable wait-states/ready

- **RAM**
  - A 16 bit address bus and an 8 or 16 bit read or write data bus

Byte addressable/writable
programmable wait-states/ready

- **AUX**
  - A 16 bit address bus and an 8 or 16 bit read or write data bus
  - Byte addressable/writable
  - programmable wait-states/ready

- **ZV output port**
  - Sync outputs (hsync & vsync)
  - Data valid indicator
  - 8 bits of Y (luminance data)
  - 8 bits of UV (chrominance data)
  - programmable pixel clock output

- **General Purpose Input/Output (GPIO)**
  - 4 general purpose I/O pins
  - Programmable direction and polarity

- **Miscellaneous Signals**
  - Local bus clock output
  - Reset output
  - Interrupt input
  - External ready input

The local bus operational configuration shall be programmable via control registers specified in section 6.2.16 page 72 of this specification.

5.2.1.5.1 RPL ROM Interface

The Remote Program Load (RPL) ROM provides the host system with the capability of reading boot code from an attached RPL ROM. This allows the system to boot from a 1394 device, even though the system may lack specific 1394 boot code at power-reset.

Additionally, this interface has been generalized to provide functionality beyond RPL ROM access. This interface will support PCI slave and internal DMA machine read/write access to devices such as EEPROM, FLASH, and other RAM-like devices.

ROM access is controlled by the standard PCI configuration RPL ROM base address register (offset 0x30) and is enabled by writing a 1 to the LSB of this register.

The ROM interface may be configured as either 8-bit or 16-bit wide data, a specified number of wait-states or external ready paced. ROM options are configured at power-reset via the serial EEPROM.

ROM Control Register [7:0]		
Bit No.	Bit Name	Description
07-04	ROM_WS[3:0]	# of wait states (0 through 14), 1111 = Pace transfer based on AUX_RDY w/timeout
02-03	reserved	Returns 0 when read
01	ROM_WR_EN	Write Enable (writable non-volatile memory)
00	ROM_16/8	Data Width, 1= 16 bit data, 0= 8 bit data.

5.2.1.5.2 SRAM Interface

The Static RAM is accessed through a second PCI memory base address register (offset 0x14). This memory may be used for DMA control structures or data buffers or a shared memory interface to other functions such as a DSP.

The RAM interface may be configured as either 8-bit or 16-bit wide data, a specified number of wait-states or external ready paced.

RAM Control Register [15:8]		
Bit No.	Bit Name	Description
15-12	RAM_WS[3:0]	# of wait states (0 through 14), 1111 = Pace transfer based on AUX_RDY w/timeout
11-09	reserved	Returns 0 when read
08	RAM_16/8	Data Width, 1= 16 bit data, 0= 8 bit data.

5.2.1.5.3 AUX Interface

This generic I/O port is accessed through a third PCI memory base address register (offset 0x18). This port may be used to implement a high speed data path to external dedicated resources such as compression/decompression logic, or video processor/frame buffers.

If the ZV Port is enabled, addresses between 0xF000 and 0xFFFF are mapped to ZV Port space; otherwise, this space is available as part of the AUX address space.

The AUX interface may be configured as either 8-bit or 16-bit wide data, a specified number of wait-states or external ready paced.

AUX Control Register [23:16]		
Bit No.	Bit Name	Description
23-20	AUX_WS[3:0]	# of wait states (0 through 14), 1111 = Pace transfer based on AUX_RDY w/timeout
19	reserved	Returns 0 when read
18	AUX_INT_POL	Interrupt polarity, 1= low true interrupt, 0=high true interrupt
17	AUX_RSTZ	AUX port reset output (low_true)
16	AUX_16/8	Data Width, 1= 16 bit data, 0= 8 bit data.

5.2.1.5.4 ZV Interface

The Zoom Video (ZV) port is an output-only port designed to transfer data from 1394 video devices to an external device on the LYNX's ZV port. When correctly programmed, this interface provides a method to receive 1394 Digital Camera packets and transfer the payload data to an external ZV-compliant device. The ZV port assumes quadlet data.

This port is accessed via a subset of the third PCI memory base address register (offset 0x18). When the ZV port is enabled, AUX addresses between 0xF000 and 0xFFFF map into the ZV Port. The ZV port is enabled when 1 of 6 available clock sources is selected as the ZV pixel clock. If none of the 6 are selected, ZV is disabled and AUX claims the entire address space. When the ZV port is disabled, all ZV-related outputs are tri-stated with the exception of the data bus which will
still be driven during AUX, RAM, & ROM accesses.

A vertical sync is generated on detection of the 1394 header sync field bit 24 (little endian). Upon detection of this sync bit, a vertical sync output is generated. For the remainder of the frame, a horizonal sync output is generated whenever a special address is used for the PCL data buffer address. By properly programming the PCL, all 1394 Digital Camera packets may be transferred via the ZV port.

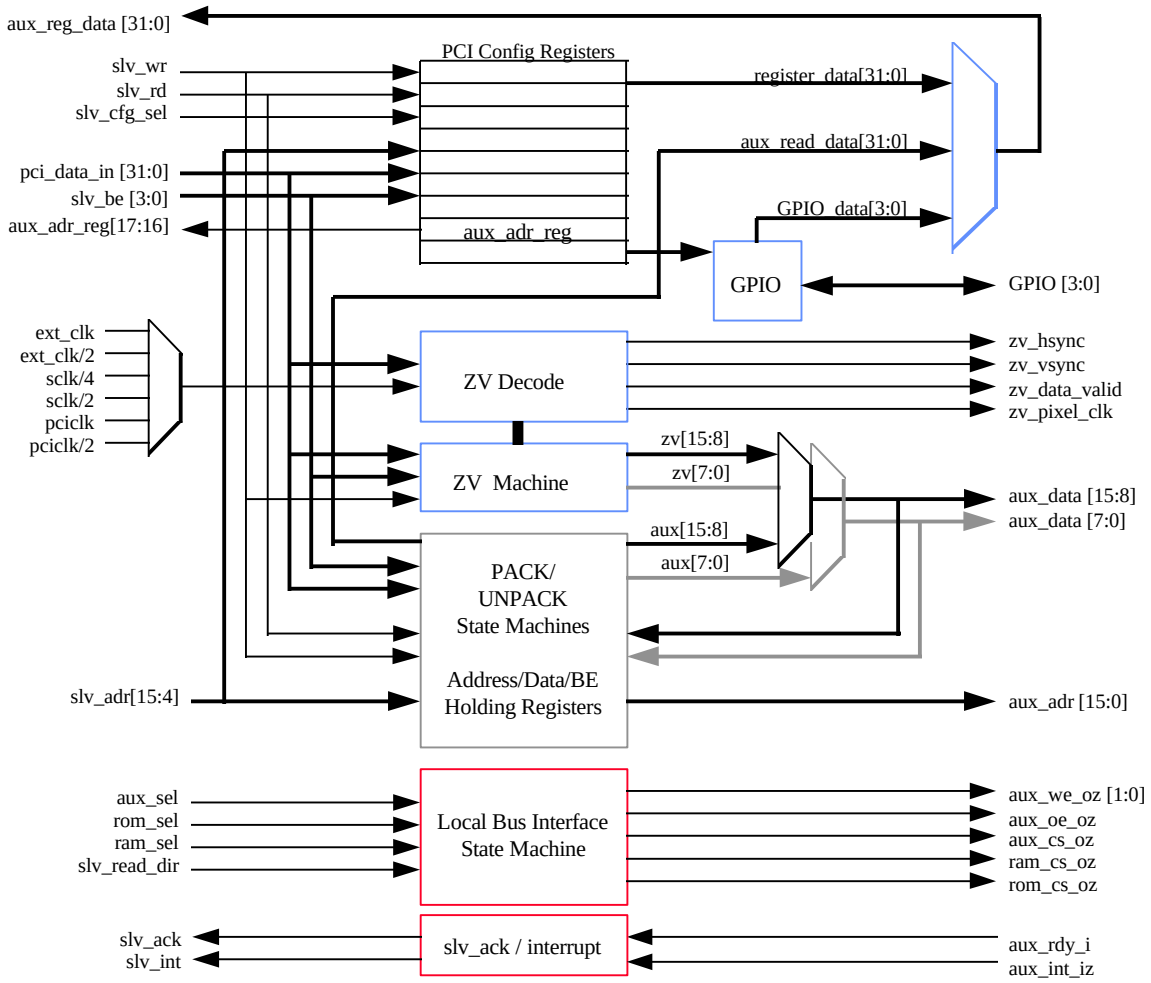
ZV Control Register [31:24]		
Bit No.	Bit Name	Description
31	CLK_GATE_ENBL	Gate ZV pixel clock, 1=gate pixel clock, 0=free running
30-28	HSYNC_CNT[2:0]	Horizontal sync count (Hsync count of 0 will still produce an hsync every frame, i.e. during vsync)
27-25	ZV_CLK[2:0]	ZV pixel clock select
24	ZV_16/8	Data Width, 1= 16 bit data, 0= 8 bit data.

5.2.1.5.5 GPIO Interface

The General Purpose I/O (GPIO) port consists of 4 general-purpose output ports. The operating mode of these 4 ports are independent and fully S/W programmable via two 32 bit control registers (16 bits per GPIO port). GPIO port 0's control register bit definition is shown below.

GPIO[x] Control Register		
Bit No.	Bit Name	Description
15-13	reserved	Returns 0 when read
12-08	GPIO[x]_SRC[4:0]	data bit mux select for output on GPIO[x]
07-03	reserved	Returns 0 when read
02	GPIO_POL_OUT[x]	output polarity control (0=non-inverted, 1=inverted)
01	GPIO_POL_IN[x]	input polarity control (0=non-inverted, 1=inverted) NOTE: GPIO[2] & [3] ONLY
00	GPIO_OUT_EN[x]	output enable control (0=tristate, 1=enabled)

5.2.1.5.6 Local Bus Interface Block Diagram



Local Bus Interface Block Diagram

5.2.2 Autoboot Mode Option

When the “autoboot” pin is active (i.e. tied high), the autoboot mode is selected. The Autoboot mode enables a number of features which allow the LYNX to function autonomously:

1. After power reset, DMA channel 0 will fetch the address of the first PCL from address 0x00000000.

2. After power reset, DMA master access to the external RPL ROM is enabled, with its base address set to 0x00000000.

3. After power reset, DMA master access to internal LYNX registers is enabled, with its internal register base address set to 0x00010000.

4. Once enabled as master on the PCI bus, the LYNX can issue PCI configuration, I/O, and memory read and write commands on the PCI bus by specifying the appropriate address range in the controlling PCL. In AUTOBOOT mode, the external PCI address space is limited to 30 bits; the two MS address bits are always zero. Internally, these 2 bits are used to select the PCI command.

AUTOBOOT = 1

Internal PCI Address

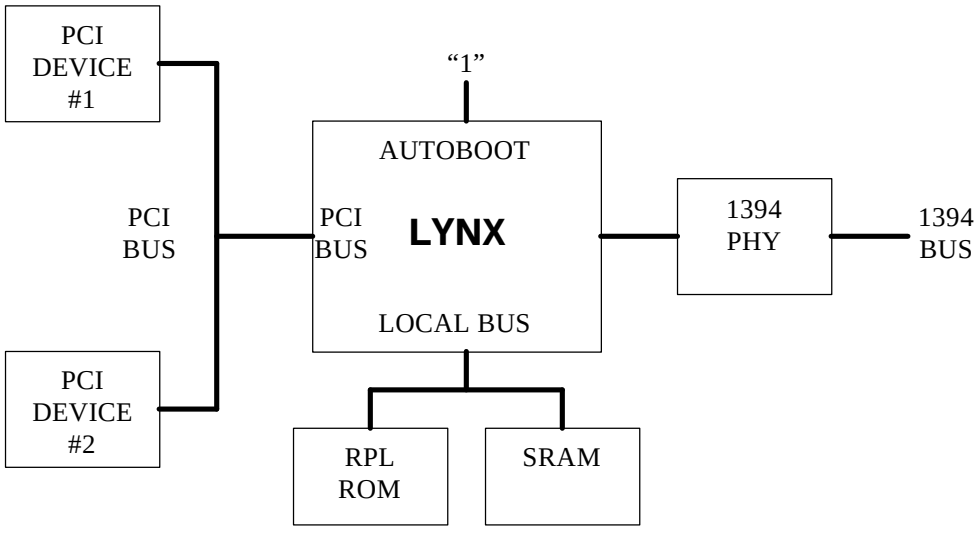
adr[31]	adr[30]	Function
0	X	PCI memory Command
1	0	PCI I/O Command
1	1	PCI Configuration Command

5. The state of the autoboot pin can be read from a special bit in the Miscellaneous control register for diagnostic purposes.

Thus, with the autoboot mode selected, and an external ROM, the LYNX can perform as the local processor to setup all the internal LYNX registers, to initialize other devices on the PCI bus, and to build and queue other PCLs. The various DMA channels can be enabled to execute these PCLs to transfer data across the 1394 bus.

By adding external SRAM to the LYNX, PCI slave memory is provided for devices on the PCI bus to obtain control information and have local memory for data transfers. PCL programs can then transfer device control/data via 1394 to another system.

This environment could be used for peripheral devices, where there may not be a suitable processor available to manage the LYNX environment. Autoboot allows this remote LYNX environment to be controlled by another agent on the 1394 bus.



5.2.3 Interrupt Logic

The interrupt logic provides control for interrupts to set the PCI bus interrupt signal INTAz from several sources. The PCI Interrupt Status register bits are each capable of generating a PCI interrupt. Any one or more status bits, when set by PCI-LYNX hardware sources, will generate a PCI interrupt and set the INT_PEND bit if the corresponding enable bit is set to 1 in the PCI Interrupt Enable. IEEE 1394-1995 status bits can also generate interrupts from the 1394 LLC Interrupt Status register bits if enabled similarly by the corresponding bit in the 1394 LLC Interrupt Enable register. If the hardware sets any one or more bits of the 1394 LLC Interrupt status register, then the P1394_INT status bit will set in the PCI Interrupt Status register.

When the hardware sets a status bit is set in either the PCI or LLC Interrupt Status registers, the status bit will set an interrupt it the corresponding enable bit is set in the PCI or LLC Interrupt Enable registers. LLC status bits require that the P1394_INT_EN bit also be set in order to generate an interrupt. Any status bit can be reset by writing a 1 to that status register bit. Once a PCI interrupt is generated, the PCI Interrupt Status register INT_PEND bit can be read to see if the interrupt was caused by the PCI-LYNX hardware. If INT_PEND is 1 (same bit can be read in either the PCI Interrupt Enable register or PCI Interrupt Status register) then one or more bits in the PCI Interrupt Status register is the source of the interrupt. Each status bit set can be cleared by writing a 1 to that bit (multiple bits can be written in one register simultaneously).

If the P1394_INT bit is set then the 1394 LLC Interrupt Status register must be read to determine which LLC status bit(s) are set. When the appropriate LLC Interrupt Status bits are cleared by writing with a 1, the LLC_INT_PEND bit in the LLC Interrupt Status register will read 0 if no new interrupt sources have occurred. Even so, the P1394_INT bit may still be set, so the P1394_INT bit must still be written with a 1 to clear P1394_INT. If an LLC Interrupt Status bit is to be polled and not interrupt enabled , then the status bit can be cleared by writing a 1, and there is no need to also write the P1394_INT bit. In other words, it is not necessary to access the PCI Interrupt Status register for any LLC Interrupt Enable bits that are always zero.

The FRC_INT bit in the PCI Interrupt Status register can be used for testing purposes. By setting the SET_FORCE_INT bit in the Miscellaneous Control register, the FRC_INT bit will be set simulating a hardware sourced condition. This feature may be useful to check out interrupt software.

Note that some interrupt conditions occur very frequently (i.e. RXDTA) during normal operation and typically should not be enabled. If all the interrupts are enabled the CPU performance could be adversely affected on typical operating systems.

5.2.4 DMA Logic

This function shall be designed to implement a 5 channel DMA controller which shall be used for transferring 1394 data packets between the host memory and the PCI-LYNX FIFO memory or between host memory and the AUX bus (refer to AUX bus description). The DMA logic shall use the PCI master logic function to acquire the PCI bus and function as a master device. The DMA logic shall be comprised of the following blocks:

- DMA engine contains a common state machine which shall be priority-time multiplexed over 5 DMA channels. This block shall also contain arbitration logic for activating a channel based on its assigned priority level.
- Control and status registers for each DMA channel along with the PCI slave data path control for accessing these registers from the PCI interface.

The DMA is controlled by data structures called Packet Control Lists or PCLs. The PCL contains command information which the DMA fetches from memory as needed. These commands tell the DMA the sources and destinations for the data and how many bytes it is to transfer. Some commands move chunks of data between the 1394 Transmit FIFOs and the PCI and between the General Receive FIFO or GRF and the PCI. Another command moves data between the PCI and the AUX bus. Other commands are for secondary functions and are called auxiliary commands. These auxiliary commands allow the DMA to peek and poke quadlets of specified data to any PCI address and permit some conditional branching (described in the section defining PCL queues). The intended use is to permit the DMA to function as a standalone processor which can build PCLs during an autoboot sequence (refer to the aforementioned autoboot description). The entire scope of this functionality is not regimented and further uses will evolve over time as programming ensues.

Each DMA channel can execute several commands. A group of commands deal with moving data between the 1394 bus and host memory. These transfer commands are XMT, UNFXMT, RCV, and RCV AND UPDATE. Only one channel of the DMA be assigned to asynchronous transmits due to complications arising from retries on the 1394 bus. Two more commands deal with moving data between the PCI bus and the auxiliary bus.
These transfer commands are called:

- XMT Transmit data from host memory to the 1394 bus.
- UNFXMT Same as XMT except that 'unformatted' data is passed through the Link Layer Controller (LLC). This permits transmission of data with the CRC as part of the payload data and not generated and appended by the LLC.
- RCV Receive data from the 1394 bus and transfer it to host memory.
- RCV_AND_UPDATE Same as RCV except that the remaining PCL transfer count and next data buffer address for the current scatter table entry is returned to the PCL in offsets 0x10 and 0x14.
- LBUS_TO_PCI Move a block of data from Local bus to host memory. The beginning local bus address is specified by the contents of the Local Bus Address Register and the beginning PCI address is specified by the Data Buffer Address Pointer in the PCL.
- PCI_TO_LBUS Move a block of data from host memory to the Local bus. The beginning local bus address is specified by the contents of the Local Bus Address Register and the beginning PCI address is specified by the Data Buffer Address Pointer in the PCL.

Seven other commands deal with branching and synchronization between DMA channels if one wishes to do so. These auxiliary commands are:

- LOAD @SOURCE => TEMP
  Read quad data from a source address and store it in a DMA temp location. The intended use is to permit a DMA channel to queue another DMA channel.
- STORE_QUAD 4 bytes from TEMP => @DESTINATION
  Move data from a DMA temp location to an address. The intended use is to permit a DMA channel to queue another DMA channel.
- STORE_DOUBLE 2 bytes from TEMP => @DESTINATION
  Move data from a DMA temp location to an address. The intended use is to permit a DMA channel to move a remaining transfer count from one PCL to the scatter table transfer count field of another PCL without overwriting the command bits of the destination PCL.

- STORE0 00000000 => @DESTINATION
  Write all zeros to an address. The intended use is to allow one DMA channel to inform a waiting DMA channel to continue execution.
- STORE1 FFFFFFFF => @DESTINATION
  Write all ones to an address. The intended use is to allow one DMA channel to inform a waiting DMA channel to continue execution.
- BRANCH DESTINATION => NEXT PCL ADDRESS if Condition True
  Conditional branch. Used to alter channel execution.
- COMPARE Compare the current contents of the TEMP register to a 16 bit immediate value with a 16 bit mask and store the equality result in the Ready register.
- ADD Add the current contents of the Temp register to a 3 bit immediate value and store the results back into the Temp register.

Figure 3: DMA channel to 1394 transfer Mode Assignments

DMA channel	Transfer Mode	DMA transfer direction
0 through 4	receive 1394 ISO / Async data transmit 1394 ISO data transmit 1394 async data * transfer data from PCI bus to LOCAL bus transfer data from LOCAL bus to PCI bus Auxiliary commands	From GRF FIFO to Host memory. From Host memory to ISO transmit FIFO. From Host memory to ASYNC transmit FIFO. ADD,BRANCH,COMPARE,LOAD,NOP, STORE,STORE0,STORE1,STORE_DOUBLE

* NOTE: Only one DMA channel shall be programmed for asynchronous transmits.
The application software shall program the operation of a DMA channel by using a Packet Control List (PCL) data structure. This structure shall reside in host memory. Application software shall be responsible for constructing the PCLs and allocating memory for their storage. A PCL shall be organized as a contiguous set of memory locations, that shall contain the commands, control parameters, and data buffer pointers required by a DMA channel to transfer one 1394 data packet, to move data between the PCI bus and auxiliary bus, or to execute one or more auxiliary commands. The total number of memory locations required to construct a PCL shall be limited to 32 quadlets. As a minimum requirement, the PCL starting address shall be aligned to a quadlet boundary (2 address lsb's = 00). For optimal DMA performance, the PCL start address is recommended to be aligned on a cache line boundary. The data buffer pointers shall as a minimum requirement, be address aligned on a byte boundary. For optimum DMA performance, it is recommended to align data buffer pointers on a cache line boundary. If this is not possible, then align to a quadlet boundary. The sum of the sizes of the data buffers pointed to by the PCL, shall not exceed 1k bytes (1394 bit rate=100mbps) or 2k bytes (1394 bit rate=200mbps). The active DMA channel shall fetch the commands and control parameters from the PCL, and use them to configure itself to perform the command or transfer. The format of a PCL for the transfer commands is defined in **Figure 4: 1394 TRANSFER Packet Control List Format.** The format of a PCL for auxiliary commands is defined in **Figure 5: AUXILIARY Command Packet Control List Format.**

Figure 4: 1394 TRANSFER Packet Control List Format

offset	PCL contents		DMA channel access performed
0x0	Next PCL address		read
0x4	Next PCL address after an Async retry transmit overrun or a Async transmit target timeout .		read
0x8	Reserved for use by software		ignored
0xC	PCL status and total transferred count		Updated by the DMA upon completion of PCL
0x10	Remaining Transfer count for the current scatter table entry.		Updated by the DMA upon completion of PCL for RCV AND UPDATE commands. Ignored by the DMA for other commands.
0x14	Next Data Buffer address for the current scatter table entry.		Updated by the DMA upon completion of PCL for RCV AND UPDATE commands. Ignored by the DMA for other commands.
0x18	PCL command	Data buffer0 control and byte count	read.
0x1C	Data buffer0 address pointer		read
0x20	Data buffer1 control and byte count		read
0x24	Data buffer1 address pointer		read
0x28	Data buffer2 control and byte count		read
0x2C	Data buffer2 address pointer		read
⋮	⋮		⋮
0x78	Data buffer12 control and byte count		read
0x7C	Data buffer12 address pointer		read

Next PCL address offset 0x0		
Bit No.	Bit Name	Description
31-04		Address of Next PCL
03-02		These bits should be = 00 for maximum performance
01	0	The PCL is to be written on a 32 bit boundary, so this bit must be 0
00	Not Vld	Next PCL address Valid. 1= Not Valid, the DMA will pause after execution of this PCL . Resumption of the DMA is caused by writing a valid Next PCL address and setting the Link bit of the DMA Control register. 0= Valid, the DMA will chain to the PCL pointed to by bits 31-02 and continue.

Next PCL Stream offset 0x4 **Note: This PCL address points to an alternate PCL queue that the active DMA channel will switch to if a 1394 retry transmit packet timeout causes the channel to become blocked on the current PCL queue.**		
Bit No.	Bit Name	Description
31-04		Address of Next PCL
03-02		These bits should be = 00 for maximum performance
01	0	The PCL is to be written on a 32 bit boundary, so this bit must be 0
00	Not Vld	Next PCL address Valid. 1= Not Valid, the DMA will pause after execution of this PCL . Resumption of the DMA is caused by writing a valid Next PCL address and setting the Link bit of the DMA Control register. 0= Valid, the DMA will chain to the PCL pointed to by bits 31-04 and continue.

Reserved for Software use offset 0x8		
Bit No.	Bit Name	Description
31-00		This word is ignored by the DMA. Software may, for example, use these locations for flags or pointers to the previous PCL in a PCL queue.

<table>
<tr><th colspan="4">Status and transferred count
offset 0xC</th></tr>
<tr><td>Bit No.</td><td>Bit Name</td><td colspan="2">Description</td></tr>
<tr><td>31</td><td>reserved</td><td colspan="2">Written with unknown data by the DMA.</td></tr>
<tr><td>30</td><td>ISO MODE</td><td colspan="2">The Received Packet was an ISO Packet.</td></tr>
<tr><td>29</td><td>Mst Err</td><td colspan="2">PCI Master Error. Set to a 1 by the DMA if it receives an error indication (parity error, timeout, etc.) from the PCI Master during execution of this PCL. In general this is a fatal condition which will cause the channel to stop, the LINK, BSY, and ENA bit are cleared in the DMA Command register (see register definitions) and an DMA_HLT interrupt (see Interrupt Status Register)
will be generated if enabled.</td></tr>
<tr><td>28</td><td>Pkt Err</td><td colspan="2">Packet Error. Set to a 1 by the DMA for any transfer to or from the 1394 bus in which the transfer had an error. The error can be determined from the Ack_Type and Acks fields. Pkt Err may not be set if Mst Err is set since it may be impossible for the DMA to update the PCL.</td></tr>
<tr><td>27</td><td>Pkt Cmp</td><td colspan="2">Packet Complete. Written by the DMA upon completion of this packet.</td></tr>
<tr><td>26-21</td><td>Receive Dma_Cha[5:0]</td><td colspan="2">Received DMA Channel number. This is the Channel number received from the Link Controller via the receive FIFO control word. Valid only for channels programmed for receive operations. These bits shall return zeros for other commands.
<table>
<tr><td>Receive Dma_Cha[5:0]</td><td>DMA Channel Number</td></tr>
<tr><td>0 0 0 0 0 0</td><td>0</td></tr>
<tr><td>0 0 0 0 0 1</td><td>1</td></tr>
<tr><td>0 0 0 0 1 0</td><td>2</td></tr>
<tr><td>0 0 0 0 1 1</td><td>3</td></tr>
<tr><td>0 0 0 1 0 0
Others</td><td>4
reserved</td></tr>
</table></td></tr>
<tr><td>20 - 19</td><td>Rcv_Speed[1:0]</td><td colspan="2">The speed at which the packet was received for Asynchronous or Isochronous Transfers. Valid only for channels programmed for receive operations. These bits shall return zeros for other commands.
00 = 100 Mbps.
01 = 200 Mbps</td></tr>
<tr><td>18 - 15</td><td>Acks</td><td colspan="2">Packet Acknowledge. Ack status returned from the Link Layer Controller for this packet. Written by the DMA upon completion of this packet. These bits are written with zeros after completion of Auxiliary commands. These bits are written with 0x0001 after completion of an isochronous transmit or PCI to/from local bus transfers.
These bit also contain a special code for internally (non 1394) related errors when bit 14 (Ack_Type) is set. The encoding for these errors are as follows:
0000 = Link reported a Retry Overrun
0001 = Link reported a Timeout
0010 = Link reported a FIFO underrun
0101 = No expected End of receive Packet
0110 = Pipelined Async Transmit Command encountered a command other than another Async Transmit.
1110 = Link reported a corrupted header before the packet was transmitted.</td></tr>
<tr><td>14</td><td>Ack_Type</td><td colspan="2">Acknowledge type returned by 1394 Transmitter logic
Ack_Type = 0 indicates a normal 1394 ack code is returned in bits 18 - 15
Ack_Type = 1 indicates a special ack code is returned in bits 18 - 15</td></tr>
<tr><td>13</td><td>reserved</td><td colspan="2">Written with unknown data by the DMA.</td></tr>
<tr><td>12-00</td><td>Transferred Count</td><td colspan="2">For all RCV and isochronous XMT commands, the DMA will update these</td></tr>
</table>

		bits with the total number of bytes transferred for this packet. These bits are indeterminate for asynchronous transmits due to the potentially pipelined nature of asynchronous XMT commands. These bits are written with zeros after completion of auxiliary commands.

Remaining Transfer Count offset 0x10		
Bit No.	Bit Name	Description
31-13	0	Written with zeros by the DMA for the RCV_AND_UPDATE command
12-00	Remaining count	For a RCV_AND_UPDATE Command, these bits will be updated by the DMA with the remaining transfer count for whatever scatter table Control,Byte Count(n) the DMA was last using when the end of the incoming receive packet was encountered. The intention is to provide this information for receiving packet data into a contiguous receive buffer.

Next Buffer Address offset 0x14		
Bit No.	Bit Name	Description
31-00	Next Buffer Address	For a RCV_AND_UPDATE Command, these bits will be updated by the DMA with the next address of whatever scatter table Data Buffer (n) the DMA was last using when the end of the incoming receive packet was encountered. The intention is to provide this information for receiving packet data into a contiguous receive buffer. See APPENDIX F.1 - TRANSFER “AT ADDRESS” PROGRAM for a description.

Transfer Command Data Buffer0 Control, Byte Count offset 0x18						
Bit No.	Bit Name	Description				
31-28	reserved	These bits shall be set to all zeros.				
27-24	CMD3-0	CMD3	CMD2	CMD1	CMD 0	Command
		0	0	0	1	RCV. (1394 FIFO to memory)
		1	0	1	0	RCV_AND_UPDATE
		0	0	1	0	XMT. (Memory to 1394 FIFO)
		1	1	0	0	UNFORMATTED XMT
		1	0	0	0	PCI_TO_LBUS
		1	0	0	1	LBUS_TO_PCI
		A packet control list queue must have consistent RCV or XMT commands. I.E. the transfer direction must be consistent.				
23	reserved	This bit shall be set to zero.				

<table>
<tr><td>22-20</td><td>Wait Sel 2-0</td><td colspan="4">Wait Select. Written when the PCL is built. These bits control what conditions have to be met before execution of the PCL will continue.</td></tr>
<tr><td></td><td></td><td>Wait Sel 2</td><td>Wait Sel 1</td><td>Wait Sel 0</td><td>Wait Condition</td></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>0</td><td>No Wait. Continue execution.</td></tr>
<tr><td></td><td></td><td>0</td><td>0</td><td>1</td><td>Wait for DMA Ready Register = 1</td></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>0</td><td>Wait for DMA Ready Register = 0</td></tr>
<tr><td></td><td></td><td>0</td><td>1</td><td>1</td><td>Wait for External Ready pin RDY = 1</td></tr>
<tr><td></td><td></td><td>1</td><td>0</td><td>0</td><td>Wait for External Ready pin RDY = 0</td></tr>
<tr><td></td><td></td><td>1</td><td>0</td><td>1</td><td>Wait for GPIO port 2 to go active</td></tr>
<tr><td></td><td></td><td>1</td><td>1</td><td>0</td><td>Wait for GPIO port 3 to go active</td></tr>
<tr><td></td><td></td><td colspan="3">Others</td><td>Reserved, do not use.</td></tr>
<tr><td>19</td><td>Int</td><td colspan="4">Generate an Interrupt. Written when the PCL is built. Is read by the DMA to determine if an interrupt is to be posted when the DMA completes updating the status for this PCL. An interrupt will be generated by the DMA regardless of the state of this bit in the case of an error which results in a termination of PCL execution by the DMA.</td></tr>
<tr><td>18</td><td>Last Buf</td><td colspan="4">Last Buffer indicator. Written when the PCL is built. Is read by the DMA to determine the end of a packet during transmits or the ending buffer for a PCI_TO_LBUS or LBUS_TO_PCI transfer.</td></tr>
<tr><td>17</td><td>Wait for Status</td><td colspan="4">Written when the PCL is built. Is used to ‘single thread’ asynchronous transmits. Normally, transmits of asynchronous transmits are pipelined to improve throughput. Setting this bit will cause the DMA to wait for transmit completion status before continuing. This bit must be set in any asynchronous transmit PCL that precedes a PCL with an auxiliary command.</td></tr>
<tr><td>16</td><td>Big Endian</td><td colspan="4">Byte ordering. Written when the PCL is built. Is read by the DMA to control the byte ordering of the data buffer as it is read or written. This bit is only used for RCV and XMT commands. It is ignored by the DMA at other times. NOTE: The Big Endian flag may only be changed on quadlet boundaries. I.E. between header and payload data.
0=Little Endian (3,2,1,0)
1=Big Endian (0,1,2,3)</td></tr>
<tr><td>15-14</td><td>xmt_spd_code[1:0]</td><td colspan="4">1394 transmit speed code. Specifies the transmission speed of an aynchronous or isochronous transmit packet. xmt_spd_code[1:0] = 00 - 100mbps
xmt_spd_code[1:0] = 01 - 200mbps
The value of this field is only valid for DMA transmit commands.</td></tr>
<tr><td>13</td><td>Multi ISO Packet per Cycle Start</td><td colspan="4">Written when the PCL is built. This bit is relevant for an isochronous Transmit DMA channel (ISO Mode = 1).
0=This isochronous packet should be sent with regard to cycle start bus boundaries. One isochronous packet per isochronous DMA channel per cycle start period.
1=This isochronous packet should be sent without regard to cycle start bus boundaries. This implies multiple isochronous packets for the same DMA channel may be transmitted during a cycle start period.</td></tr>
<tr><td>12</td><td>Transmit ISO Mode</td><td colspan="4">Written when the PCL is built. If the command specified by bits 24-27 is a 1394 transmit, then:
0= This DMA channel is to be configured for Transmit asynchronous transfers.
1=This DMA channel is to be configured for Transmit isochronous transfers.</td></tr>
<tr><td>11-00</td><td>Transfer Count</td><td colspan="4">Data Buffer Transfer Length in bytes. Written when the PCL is built. Is read by the DMA to determine the size of this buffer.</td></tr>
</table>

Transfer Command Data Buffer (0 to n) Address Pointer offset 0x1C, 0x24... etc.		
Bit No.	Bit Name	Description
31-00	DATA_BUF	Address of this Data Buffer. This address may begin on any byte boundary but for maximum PCI transfer rates this address should begin on a cache line size boundary. For RCV and XMT commands this represents the address of host data. For LBUS_TO_PCI and PCI_TO_LBUS transfers, this represents the address of the PCI bus data. The address of the LOCAL bus source or destination is contained in the LOCAL bus base address register.

Transfer Command Data Buffer (1 to n) Control, Byte Count offset 0x20, 0x28, etc.		
Bit No.	Bit Name	Description
31-19	reserved	These bits shall be set to all zeros.
18	Last Buf	Last Buffer indicator. Written when the PCL is built. Is read by the DMA to determine the end of a packet during transmits or the ending buffer for a PCI_TO_LBUS or LBUS_TO_PCI transfer.
17	reserved	These bits shall be set to all zeros.
16	Big Endian	Byte ordering. Written when the PCL is built. Is read by the DMA to control the byte ordering of the data buffer as it is read or written. This bit is only relevant for transfers between the 1394 bus and host memory. NOTE: The Big Endian flag may only be changed on quadlet boundaries. I.E. between header and payload data. 0=Little Endian (3,2,1,0) 1=Big Endian (0,1,2,3)
15-12	reserved	These bits shall be set to all zeros.
11-00	Transfer Count	Data Buffer Transfer Length in bytes. Written when the PCL is built. Is read by the DMA to determine the size of this buffer.

Figure 5: AUXILIARY Command Packet Control List Format

offset	PCL contents	DMA channel access performed
0x0	Next PCL address	read
0x4	unused	ignored
0x8	Reserved for use by software	ignored
0xC	PCL status	Updated by the DMA upon completion of PCL
0x10	unused	ignored
0x14	unused	ignored
0x18	PCL auxiliary command 1	read.
0x1C	parameter for auxiliary command 1	read
⋮	Other auxiliary commands and parameters 2 - 13 (optional) ⋮	read ⋮
0x78	PCL status & auxiliary command 13 (optional)	read
0x7C	parameter for auxiliary command 13 (optional)	read

Auxiliary Command offset 0x18, 0x20... etc.						
Bit No.	Bit Name	Description				
31-28	0	These bits shall be set to all zeros.				
27-24	CMD3-0	CMD3	CMD2	CMD1	CMD0	Command
		0	0	0	0	NOP
		0	0	1	1	LOAD (@DESTINATION => TEMP)
		0	1	0	0	STORE QUAD (4 bytes TEMP => @SOURCE)
		1	0	1	1	STORE DOUBLE (2 bytes TEMP => @SOURCE)
		0	1	0	1	STORE0 (00000000 => @DESTINATION)
		0	1	1	0	STORE1 (FFFFFFFF => @DESTINATION)
		0	1	1	1	Conditional BRANCH to DESTINATION if the conditions are met as specified in the condition field. Status is updated and an interrupt is generated, if enabled, prior to the branch.
		1	1	0	1	ADD (TEMP + BUFFER => TEMP)
		1	1	1	0	Compare (TEMP ^ BUFFER => READY)
		1	1	1	1	Reserved, do not use.
		DESTINATION, @DESTINATION, and @SOURCE addresses are contained in the next word. TEMP is the DMA Previous Address register.				
23	reserved	This bit shall be set to zero.				

22-20	Wait Sel 2-0	Wait Select. Written when the PCL is built. These bits control what conditions have to be met before execution of the PCL will continue for the data movement auxiliary commands of LOAD,STORE,STORE0, and STORE1.			
		Wait Sel 2	Wait Sel 1	Wait Sel 0	Wait Condition
		0	0	0	No Wait. Continue execution.
		0	0	1	Wait for DMA Ready Register = 1
		0	1	0	Wait for DMA Ready Register = 0
		0	1	1	Wait for External Ready pin RDY = 1
		1	0	0	Wait for External Ready pin RDY = 0
		1	0	1	Wait for GPIO port 2 to go active
		1	1	0	Wait for GPIO port 3 to go active
		Others			Reserved, do not use.
22-20	Condition Codes 2-0	Branch Command Condition codes. Written when the PCL is built. These bits select what conditions have to be met during the execution of the BRANCH command to cause the address contained in DESTINATION to be loaded into the NEXT PCL ADDRESS and linked.			
		Condition Code 2	Condition Code 1	Condition Code 0	Branch Condition
		0	0	0	Don't branch
		0	0	1	Branch if DMA Ready Register = 1
		0	1	0	Branch if DMA Ready Register = 0
		0	1	1	Branch if External Ready pin RDY = 1
		1	0	0	Branch if External Ready pin RDY = 0
		1	0	1	Branch if GPIO port 2 is active
		1	1	0	Branch if GPIO port 3 is active
		Others			Reserved, do not use.
19	Int	Generate an Interrupt. Written when the PCL is built. Is read by the DMA to determine if an interrupt is to be posted when the DMA completes updating the status for this auxiliary command PCL. An interrupt will be generated by the DMA regardless of the state of this bit in the case of an error resulting in Mst Err status being set. This bit is only valid for the auxiliary command at offset 0x18.			
18	Last Command	Last Command indicator. Written when the PCL is built. Is read by the DMA to determine the last auxiliary command in a PCL.			
17-00	reserved	These bits shall be set to all zeros.			

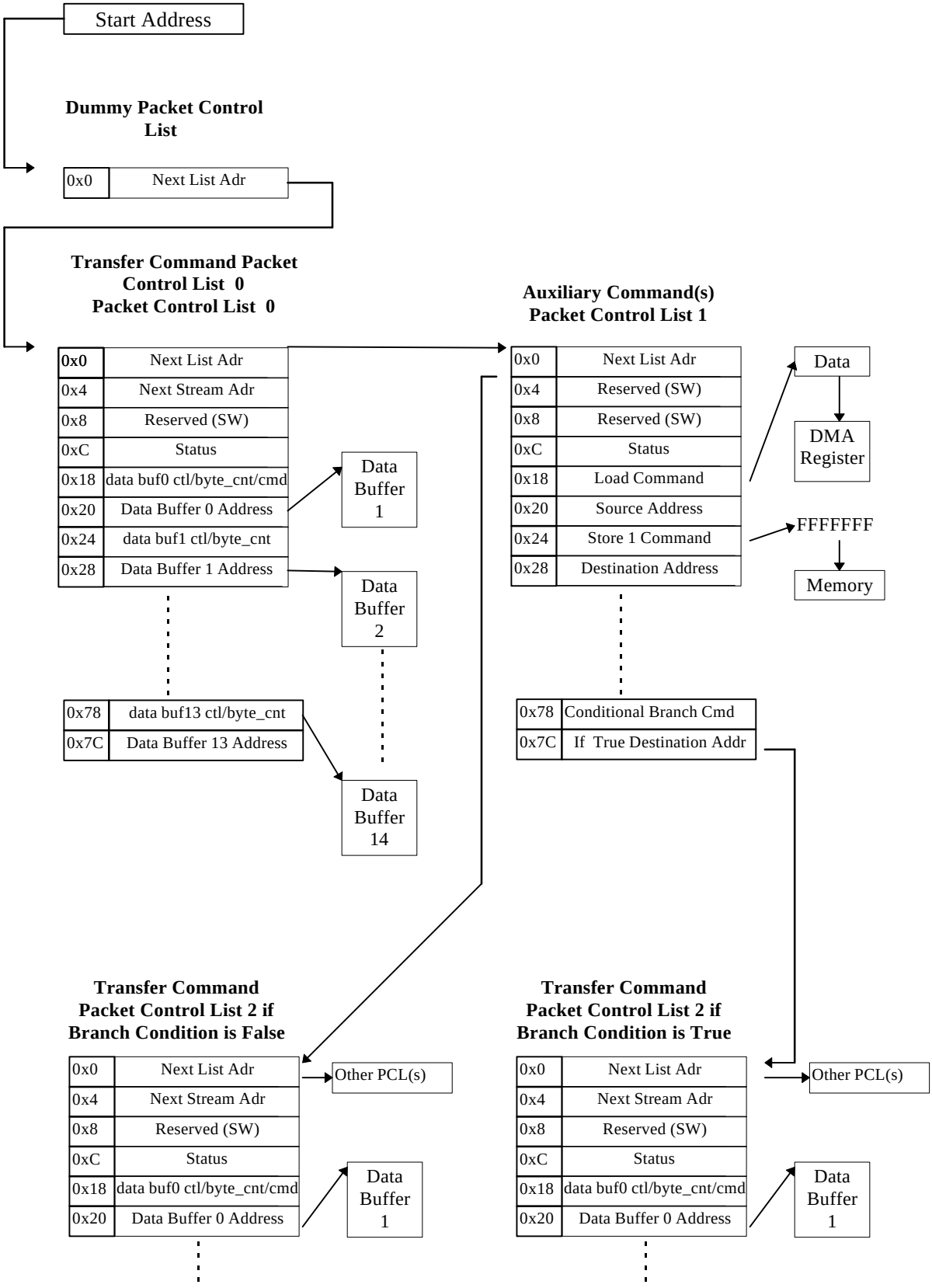
Auxiliary Command Parameter
offset 0x1C, 0x24... etc.

These bits are loaded by the DMA into the **Current Data Buffer Address Register** and are used by the DMA during the execution of the following auxiliary commands as follows:

NOP		
Bit No.	Bit Name	Description
31-00	Don't Care	Read but not used by the DMA
LOAD		
Bit No.	Bit Name	Description
31-00	@SOURCE	Address of the data that is to be stored in a temporary location in the DMA. This temporary location is the DMA's Previous Pointer/Temp Register.
STORE QUAD		
Bit No.	Bit Name	Description
31-00	@DESTINATION	Address where the data stored in the temporary location in the DMA will be written. This temporary location is the DMA's Previous Pointer/Temp Register.
STORE DOUBLE		
Bit No.	Bit Name	Description
31-00	@DESTINATION	Address where the data stored in the temporary location in the DMA will be written. This temporary location is the DMA's Previous Pointer /Temp Register.
STORE0		
Bit No.	Bit Name	Description
31-00	@DESTINATION	Address where data of 00000000 will be written.
STORE1		
Bit No.	Bit Name	Description
31-00	@DESTINATION	Address where data of FFFFFFFF will be written.
COMPARE		
Bit No.	Bit Name	Description
31-16	Compare Enable	Each bit set to 1 here will enable the corresponding bit compare in bits 15-00 respectively. Each bit set to 0 here will mask the the corresponding bit compare in bits 15-00 respectively.
15-00	Compare Value	This value is bit-wise compared against bits 15-00 respectively of the current contents of the DMA's Previous Pointer /Temp Register. The logical result (1=equal, 0=not equal) is written to the Ready Register's bit 0.
ADD		
Bit No.	Bit Name	Description
31-03	Don't Care	Read but unused
02-00	Addend	Added to the current 32 bit contents of the DMA Previous Pointer /Temp Register and the result stored back into the Previous Pointer /Temp Register.
BRANCH		
Bit No.	Bit Name	Description
31-00	DESTINATION	This address is loaded into the CURENT PCL ADDRESS if the conditions are met as specified in the condition code field.

Application software shall program a DMA channel to transfer multiple 1394 data packets by chaining together multiple PCLs into a Packet Control List Queue. A queue shall be constructed by setting the next address field of each PCL, to point to the starting address in memory of the next PCL. The Last PCL in the queue can be programmed to either halt DMA processing, point back to the start of the queue, or point to a new queue. See **Figure 6: Example PCL Queue**. PCLs containing auxiliary command(s) may be embedded anywhere in a PCL queue but a given PCL may only contain transfer commands or auxiliary commands but not both. A PCL queue may mix RCV and XMT and auxiliary commands together
however, an asynchronous transmit command must be followed by another asynchronous transmit commands due to the potential piplined nature transmits and the possibility of a packet retry. Setting the "Wait for Status" bit in the command word of an asynchronous transmit precludes this requirement.

Figure 6: Example PCL Queue



5.2.4.1 DMA Engine

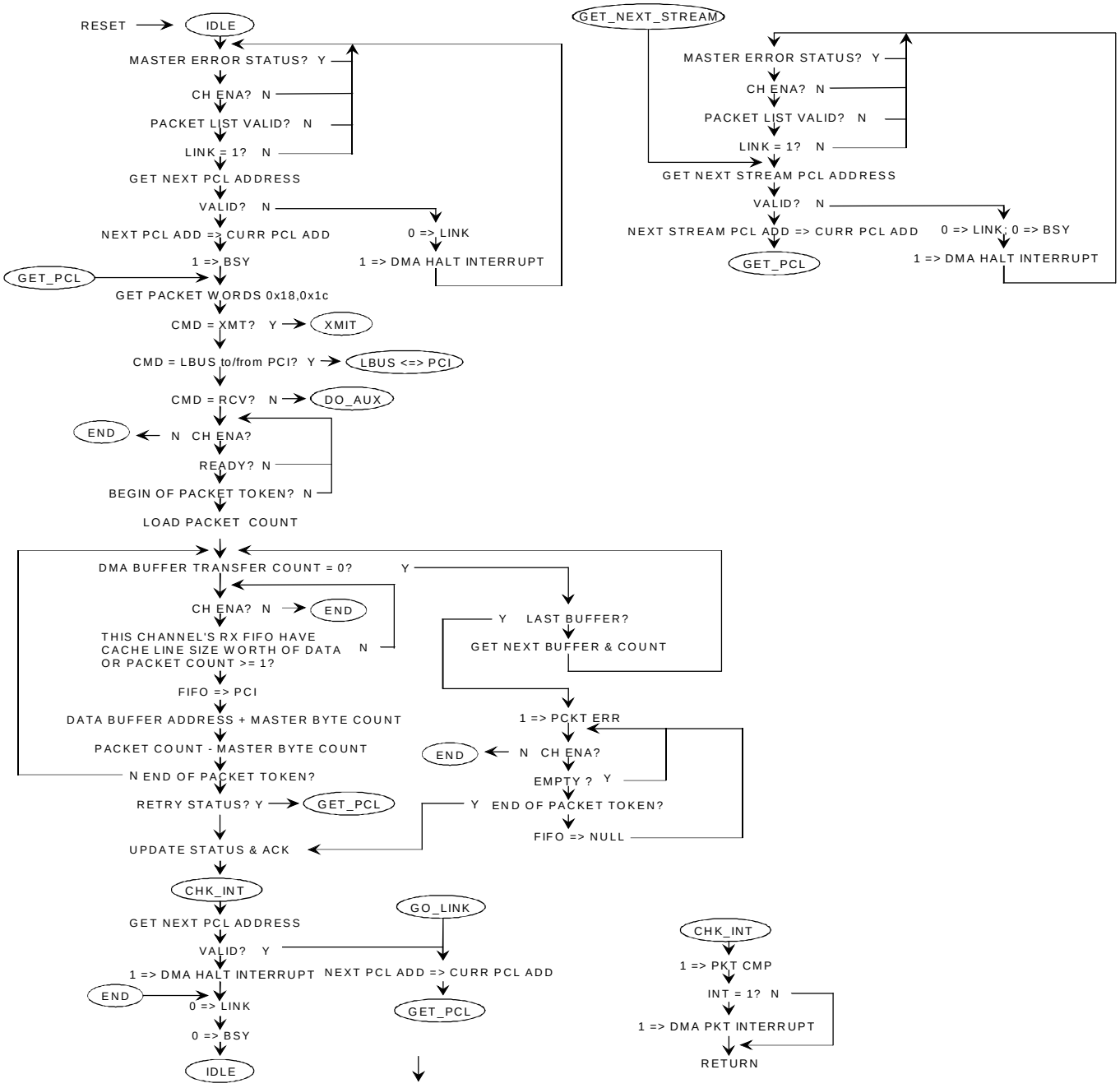
The DMA engine function shall implement the state machine logic for fetching the control parameters and data buffer pointers from the PCL. The state machine logic or packet processor shall use these parameters to control the transfer of data to or from the data buffers. The DMA channel priority assignment is as defined in **Figure 7: DMA channel priority assignments.** The overall flow of each DMA channel is as defined in Error! Reference source not found..

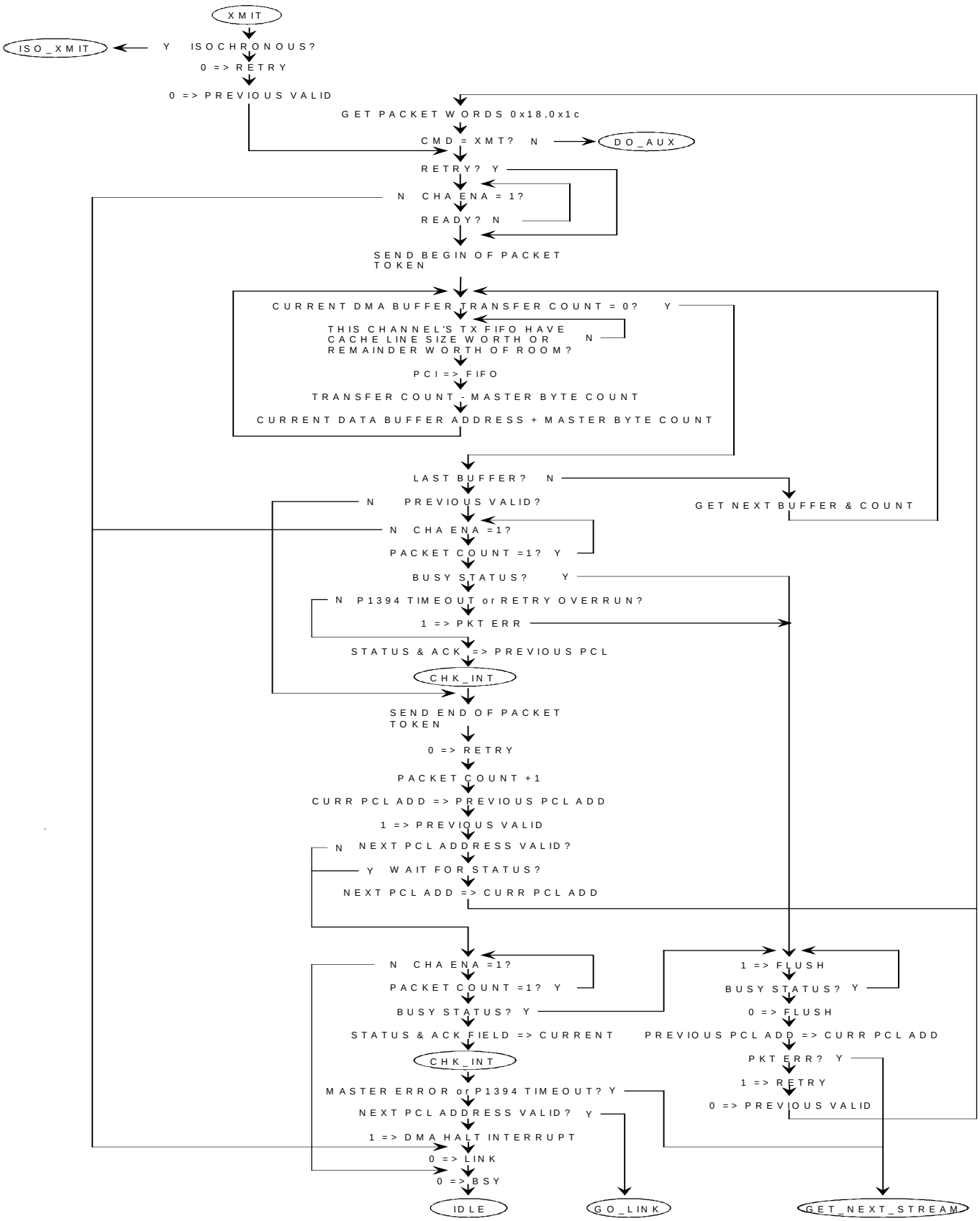
Figure 7: DMA channel priority assignments.

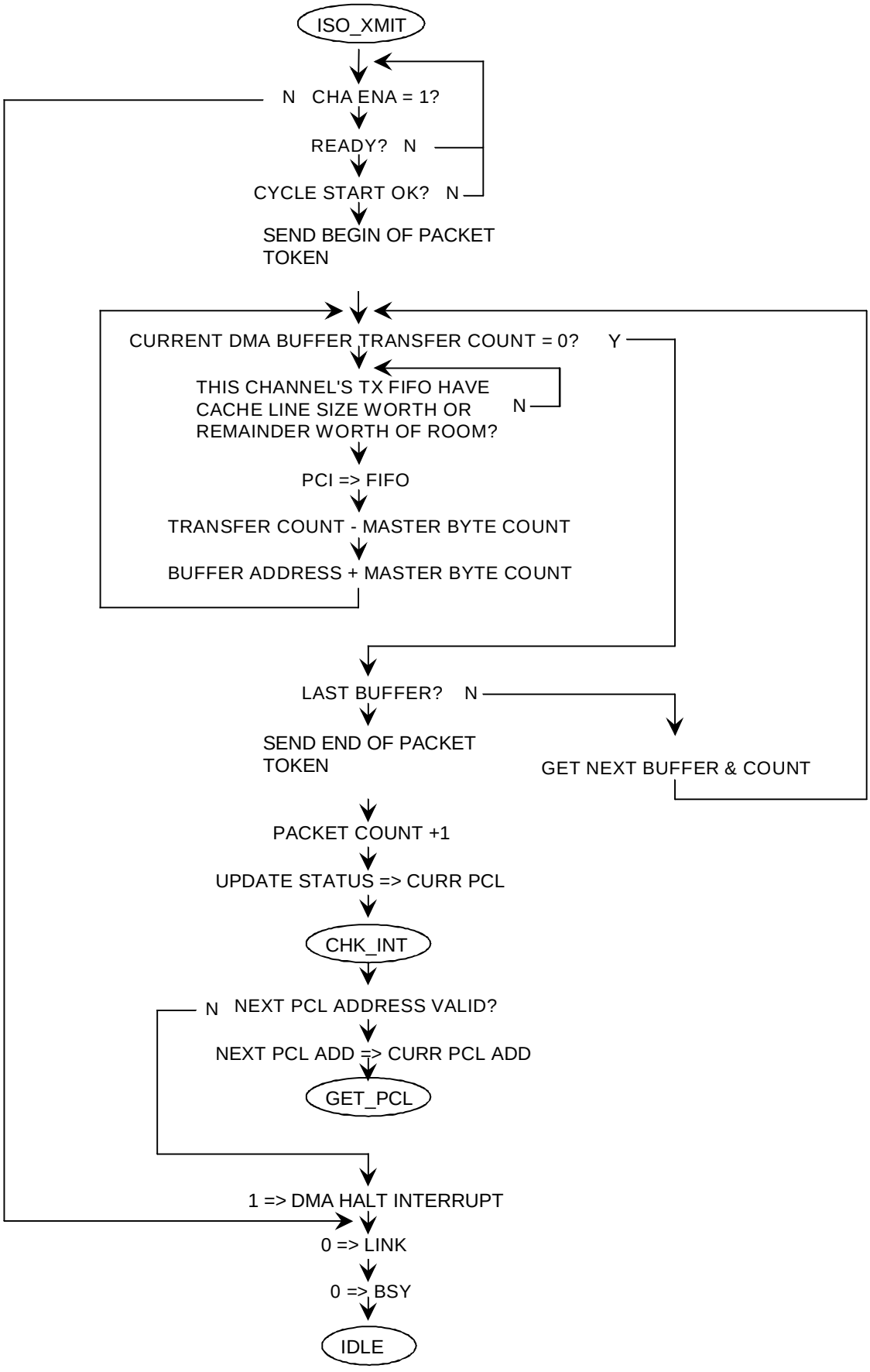
DMA channel	priority	1394 transfer type
X	(highest)	channel currently active on the 1394 bus
0		transfer commands or auxiliary commands
1		transfer commands or auxiliary commands
2		transfer commands or auxiliary commands
3		transfer commands or auxiliary commands
4	(lowest)	transfer commands or auxiliary commands

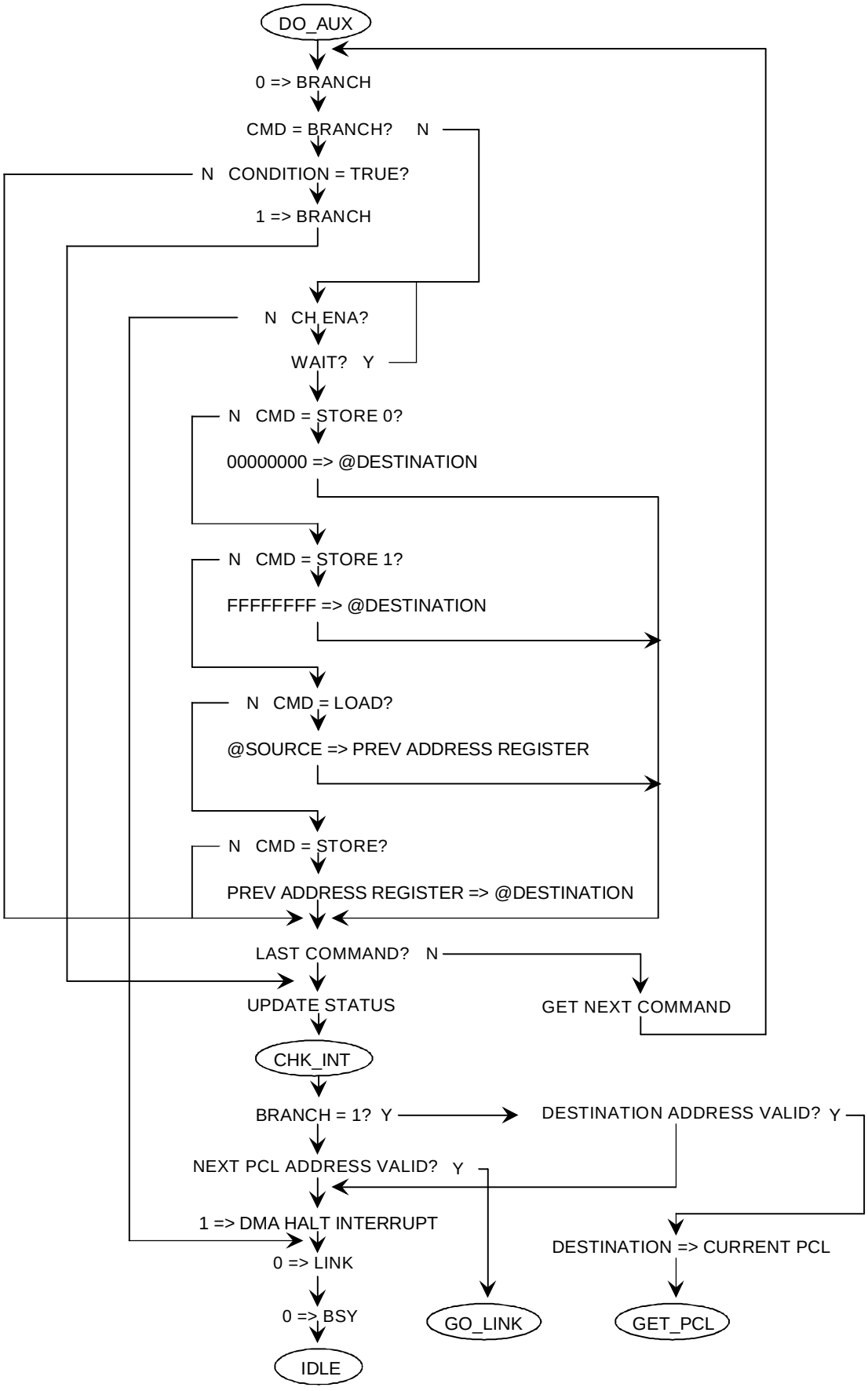
Figure 8: State Machine Control Flow Diagram

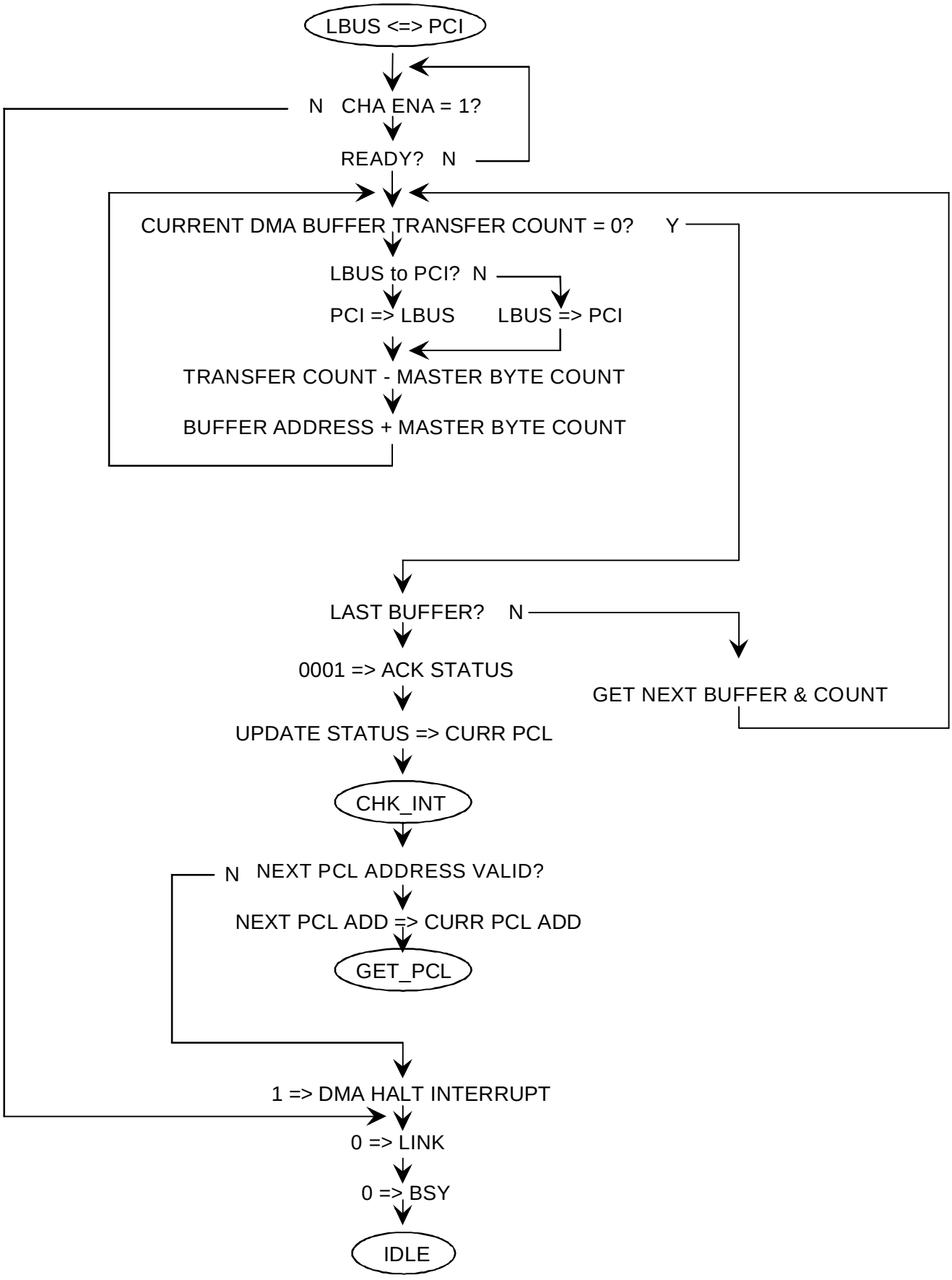
DMA PACKET PROCESSOR




XMIT
ISO_XMIT
Y ISOCHRONOUS?
0 => RETRY
0 => PREVIOUS VALID
GET PACKET WORDS 0x18,0x1c
CMD = XMT? N
DO_AUX
RETRY? Y
N CHA ENA = 1?
READY? N
SEND BEGIN OF PACKET TOKEN
CURRENT DMA BUFFER TRANSFER COUNT = 0? Y
THIS CHANNEL'S TX FIFO HAVE CACHE LINE SIZE WORTH OR REMAINDER WORTH OF ROOM? N
PCI => FIFO
TRANSFER COUNT - MASTER BYTE COUNT
CURRENT DATA BUFFER ADDRESS + MASTER BYTE COUNT
LAST BUFFER? N
GET NEXT BUFFER & COUNT
N PREVIOUS VALID?
N CHA ENA =1?
PACKET COUNT =1? Y
BUSY STATUS? Y
N P1394 TIMEOUT or RETRY OVERRUN?
1 => PKT ERR
STATUS & ACK => PREVIOUS PCL
CHK_INT
SEND END OF PACKET TOKEN
0 => RETRY
PACKET COUNT +1
CURR PCL ADD => PREVIOUS PCL ADD
1 => PREVIOUS VALID
N NEXT PCL ADDRESS VALID?
Y WAIT FOR STATUS?
NEXT PCL ADD => CURR PCL ADD
N CHA ENA =1?
PACKET COUNT =1? Y
BUSY STATUS? Y
STATUS & ACK FIELD => CURRENT
CHK_INT
MASTER ERROR or P1394 TIMEOUT? Y
NEXT PCL ADDRESS VALID? Y
1 => DMA HALT INTERRUPT
0 => LINK
0 => BSY
IDLE
GO_LINK
1 => FLUSH
BUSY STATUS? Y
0 => FLUSH
PREVIOUS PCL ADD => CURR PCL ADD
PKT ERR? Y
1 => RETRY
0 => PREVIOUS VALID
GET_NEXT_STREAM


ISO_XMIT
N CHA ENA = 1?
READY? N
CYCLE START OK? N
SEND BEGIN OF PACKET TOKEN
CURRENT DMA BUFFER TRANSFER COUNT = 0?
Y
THIS CHANNEL'S TX FIFO HAVE CACHE LINE SIZE WORTH OR REMAINDER WORTH OF ROOM?
N
PCI => FIFO
TRANSFER COUNT - MASTER BYTE COUNT
BUFFER ADDRESS + MASTER BYTE COUNT
LAST BUFFER? N
SEND END OF PACKET TOKEN
GET NEXT BUFFER & COUNT
PACKET COUNT +1
UPDATE STATUS => CURR PCL
CHK_INT
N NEXT PCL ADDRESS VALID?
NEXT PCL ADD => CURR PCL ADD
GET_PCL
1 => DMA HALT INTERRUPT
0 => LINK
0 => BSY
IDLE


DO_AUX
0 => BRANCH
CMD = BRANCH? N
N CONDITION = TRUE?
1 => BRANCH
N CH ENA?
WAIT? Y
N CMD = STORE 0?
00000000 => @DESTINATION
N CMD = STORE 1?
FFFFFFFF => @DESTINATION
N CMD = LOAD?
@SOURCE => PREV ADDRESS REGISTER
N CMD = STORE?
PREV ADDRESS REGISTER => @DESTINATION
LAST COMMAND? N
UPDATE STATUS
GET NEXT COMMAND
CHK_INT
BRANCH = 1? Y
DESTINATION ADDRESS VALID? Y
NEXT PCL ADDRESS VALID? Y
1 => DMA HALT INTERRUPT
0 => LINK
DESTINATION => CURRENT PCL
0 => BSY
GO_LINK
GET_PCL
IDLE


LBUS <=> PCI
N CHA ENA = 1?
READY? N
CURRENT DMA BUFFER TRANSFER COUNT = 0? Y
LBUS to PCI? N
PCI => LBUS
LBUS => PCI
TRANSFER COUNT - MASTER BYTE COUNT
BUFFER ADDRESS + MASTER BYTE COUNT
LAST BUFFER? N
0001 => ACK STATUS
GET NEXT BUFFER & COUNT
UPDATE STATUS => CURR PCL
CHK_INT
N NEXT PCL ADDRESS VALID?
NEXT PCL ADD => CURR PCL ADD
GET_PCL
1 => DMA HALT INTERRUPT
0 => LINK
0 => BSY
IDLE

One can think of the DMA Packet processor as 5 independent DMA channels all running concurrently. The actual implementation utilizes one main control state machine which multiplexes between the 5 DMA channels over time. Priority supervisor logic continuously examines the current context of all channels and assigns the channel with the highest priority of pending activity to the state machine for execution.

A DMA channel initializes after reset to a static condition where it is waiting for a valid PCL pointer to be written to the **Packet control list start address register** , and the **CH ENA** and **Link** bits to be set in the **DMA control register.** A valid PCL pointer is determined by the state of bit 0 of the **Packet control list start address register .** A 1 indicates an invalid address, a 0 indicates a valid address. The DMA will then go to the address pointed to by the **Packet control list start address register**, get the next address and if valid will make this the current PCL address and begin execution. If this address is invalid the **Link** bit is cleared in the **DMA control register**, a DMA halted interrupt is generated for this channel with associated status (**DMA_HLT[x]**) in the **Interrupt Status** register (see configuration register definitions) and the channel goes inactive. This mechanism provides a sanity check on the PCL memory structures as well as provides a relatively easy way to continue channel PCL execution in the event a next address link is missed. When a valid next PCL address is detected the DMA will then set the **BSY** bit in the **DMA control register**, get the words at PCL offset 0x18 and 0x20. A check is then made to determine whether the command is a receive, transmit, PCI to/from LBUS, or auxiliary command.

5.2.4.1.1 DMA Receive Operation

A receive operation for isochronous and asynchronous data in the GRF will proceed by checking to see if a wait condition exists. The wait condition is determined by the Wait Select bits of the Data Buffer Control Word 0 at PCL offset 0x18.

Once the wait condition no longer exists, the processor enters a data movement phase. Here a loop is entered where the current transfer count is checked to see if it has gone to zero. If so, a check is made to see if this is the last data buffer of the PCL buffer list. If it is the last buffer and a packet boundary has not been indicated by the Link Layer Controller writing a special control token word in the GRF FIFO, then an error has occurred because more packet data is to be transferred than the buffer can hold. In this case the **PKT ERR** bit is set in the **DMA status register** and the DMA will flush the remaining data up to the packet boundary. If the current transfer count has decremented to zero and there is another buffer in the PCL list then the DMA will acquire the new buffer address and transfer count and proceed with the transfer. While moving data from the receive FIFO to the PCI interface, the DMA will wait for the FIFO to have sufficient data before requesting the PCI bus master to perform a transfer. This transfer threshold is reached whenever one of two conditions is met. The DMA will request a transfer of the PCI master whenever the number of bytes in the receive FIFO reaches a 'high water mark'. This highwater mark is equal to the greater of the cache line size register or the lower bound field of the **DMA Global register**. The DMA gets information of a packet's data size from the link when the packet is first being written into the FIFO by the Link Layer Controller. It uses this transfer count to determine if the data in the FIFO is the remaining data in the packet and if so and the size is less than the high water mark, it will request a transfer of the PCI Master where the transfer count is equal to this remainder. While the DMA is transferring data, the **Data buffer start address register** and the **data buffer transfer length** bits in the **DMA control register** are updated to reflect the current state of the transfer.

When the Link Layer Controller encounters the end of a packet it writes a special control token word into the FIFO to mark the end of a packet. Embedded in this control word are status bits that indicate the completion state of the packet on the bus. The DMA uses this end of packet marker to terminate the transfer of data from the FIFO to the PCI bus. If the end of packet marker indicates a 1394 busy acknowledge, the DMA re-acquires the PCL's first buffer address and transfer count and starts the packet's transfer all over. If there was no busy status indicated from the end of packet marker then the **DMA status register** is loaded with the acknowledge status passed from the Link Layer Controller in the end of packet marker, the **PKT CMP** is set, and it is then written to memory in the **PCL status** word at PCL offset 0xC along with the number of bytes transferred for this PCL. If the **INT** bit is set in the **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18 then an interrupt is signaled and latched in the corresponding (**DMA_PCL[x]**) bit in the **Interrupt Status** register. If the command was a RCV AND UPDATE command then the remaining transfer count and next buffer address are written to PCL offsets 0x10 and 0x14 respectively.

The DMA then determines whether another PCL has been linked to the current PCL by fetching the **Next List Adr** (PCL offset 0x00). If it is valid as indicated by bit 0 = 0 then the DMA will make this the current PCL address and continue execution as shown. If another PCL had not been linked to the current PCL as indicated by bit 0 = 1 then the **Link** and **BSY** bits are cleared in the **DMA control register,** a DMA halted interrupt is generated for this channel with associated status (**DMA_HLT[x]**) in the **Interrupt Status** register, and the channel becomes idle.

5.2.4.1.2 DMA Asynchronous Transmit Operation

Asynchronous transmits are determined after a valid PCL pointer has been written to the **Packet control list start address register** and the **CH ENA** and **Link** bits have been set as shown in the flow chart. The overall goal of the asynchronous packet processor is to remain 1 packet ahead of the current packet being transferred from the FIFO to the 1394 bus by the Link Layer Controller. From the DMA's point of view, this packet on the bus was the previous packet. Any status reported by the Link Layer Controller is assumed to be for this previous packet however, setting the **Wait For Status** bit in the **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18 will prevent this pipelining operation. The DMA keeps the address of the previous packet control list start address in the **Previous Packet control list start address/Temp register**. A flag called 'Previous PCL Valid' is kept by the DMA in the **DMA Global register** to keep track of whether it has stored a valid address.
A transmit operation for an asynchronous channel will proceed by checking to see if a wait condition exists. The wait condition is determined by the **Wait Select** bits of the **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18. A flag called 'retry' is kept by the DMA in the **DMA Global register**. This flag is used by the DMA to keep track of when the wait conditions should be evaluated as these wait conditions are ignored during retries.

Once the wait condition no longer exists, the processor writes a control token to the FIFO indicating the beginning of a packet and enters a data movement phase. Here a loop is entered where the current transfer count is checked to see if it has gone to zero. If so, a check is made to see if this is the last data buffer of the PCL buffer list. If there is another buffer in the PCL list then the DMA will acquire the new buffer address and transfer count and proceed with the transfer. While moving data into the asynchronous transmit FIFO from the PCI interface, the DMA will wait for the FIFO to have sufficient room before requesting the PCI bus master to perform a read transfer. The DMA will request a transfer of the PCI master with the byte count equal to the high water mark as defined in the aforementioned DMA receive operation. While the DMA is transferring data, the **Data buffer start address register** and the **data buffer transfer length** bits in the **DMA control register** are updated to reflect the current state of the transfer.

When the last byte of data from a buffer has been transferred to the asynchronous transmit FIFO and the buffer is the last of the PCL list as indicated by the **LAST BUF** bit of the **ctl/byte_cnt** PCL word then the DMA knows that the end of a packet has been reached. If the previous packet address is valid, the DMA will delay checking status until there is a full packet queued in the transmit FIFO. This way returned status is always for the previous packet unless the **Wait For Status** bit is set. If there is only one packet in the transfer, then the previous and current packets are the same. If the previous packet address is valid then the DMA will look at the packet counter. When a packet has been transmitted to the 1394 bus by the Link Layer Controller and status for this packet is valid, the Link Layer Controller will decrement the packet counter. The DMA will spin waiting for packet counter to go to zero indicating valid status is available for the previous packet. If the status indicates that the previous packet is to be retried, then the DMA sets a flush FIFO request to the Link Layer Controller and then waits for the Link Layer Controller to indicate the completion of the FIFO flush by the removal of the retry indication. The DMA then "backs up to" the previous packet and starts the transfer all over. If no retry occurred, then the DMA will update the **DMA status register** with the acknowledge status passed from the Link Layer Controller, the **PKT CMP** is set, and it is then written to memory in the previous **PCL status** word at PCL offset 0xC along with the number of bytes transferred for the currently active PCL which may not be relevant for the previous PCL. If the **INT** bit is set in the **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18 then an interrupt is signaled and latched in the corresponding (**DMA_PCL[x]**) bit in the **Interrupt Status** register.

When the status has been checked, the DMA will write a special control token to the transmit FIFO to mark the end of the packet. The packet count in incremented to 1 to indicate to the Link Layer Controller that the end of packet has been written by the DMA. The current PCL address is saved as the previous PCL address in the **Previous Packet control list start address register** and a "Previous Valid" flag is set in the **DMA Global register.**
The DMA then determines whether another PCL has been linked to the current PCL by fetching the **Next List Adr** (PCL offset 0x00). If it is valid as indicated by bit 0 = 0 then the DMA will make this the current PCL address and continue

execution as shown. If it is not valid, or if the **Wait For Status** bit is set then the DMA waits for the current packet to be transferred by the Link Layer Controller. When valid status is available as indicated by the packet counter decrementing to zero, the DMA will check to see if the packet is to be retried as indicated by a 1394 busy status. If so, the FIFO is flushed as aforementioned and the transmit is attempted again.

If there was a transmit timeout, retry overrun, or FIFO underrun as indicated by the Link Layer Controller then the PKT ERR bit is set in the **DMA status register** along with the acknowledge status and the status is updated in the PCL (offset 0xC)**.** In the event of a transmit timeout or retry overrun it may be possible that the target node is no longer responding . The DMA provides for the capability in this case to skip around the PCL(s) which form the stream of data to this device. Software can set the **Next PCL Stream** entry of the PCL at offset 0x4 to point to the first PCL of the next stream of transmit data (next async transmit node). If the **Next PCL Stream** address is valid, then the DMA will continue execution with that PCL. If this address is not valid, then the DMA channel will go idle the same way as any time it encounters a next PCL address marked invalid. If this next stream feature is not to be used then one should set this entry to the same value as the **Next List Adr** (PCL offset 0x00). If the DMA halts due to **DMA_HLT[x]**, and the **Next PCL Stream** entry was invalid, then re-writing the **Next PCL Stream** entry is necessary since the DMA is in the GET_NEXT_STREAM state of **Error! Reference source not found.** and the DMA State Machine is ignoring the **Next List Adr**. Always settting the **Next List Adr** and the **Next PCL Stream** to the same address is therefore required if the next stream feature is not to be used to prevent a hang in any Async XMT channel that invokes the **Next PCL Stream** entry due to an error.

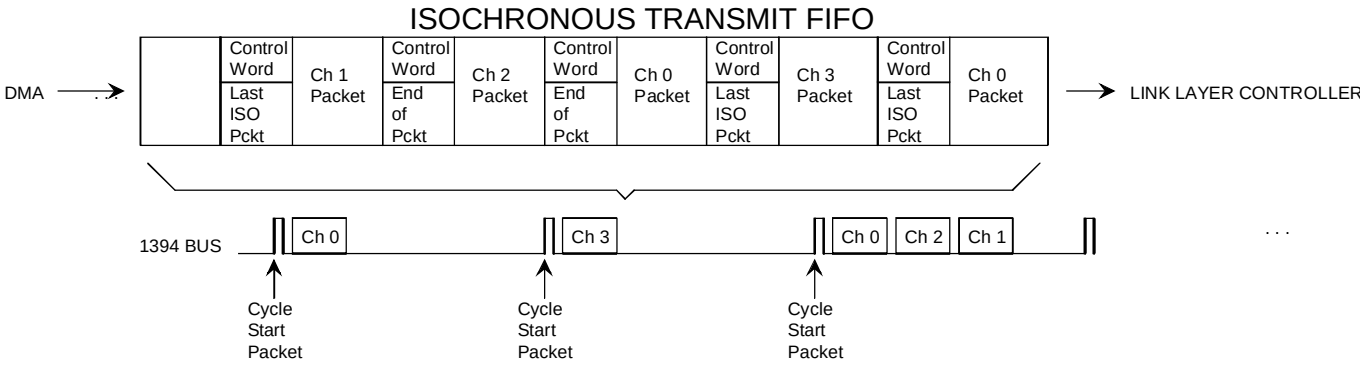
If there was no retry, timeout, or FIFO underrun the DMA will update the **DMA status register** with the 1394 acknowledge status passed from the Link Layer Controller, the **PKT CMP** is set, and it is then written to memory in the **PCL status** word at PCL offset 0xC. If the **INT** bit is set in the **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18 then an interrupt is signaled and latched in the corresponding (**DMA_PCL[x]**) bit in the **Interrupt Status** register.
If another PCL had not been linked to the current PCL as indicated by bit 0 = 1, the **Link** and **BSY** bits are cleared in the **DMA control register,** a DMA halted interrupt is generated for this channel with associated status (**DMA_HLT[x]**) in the **Interrupt Status** register, and the channel becomes idle.

5.2.4.1.3 DMA Isochronous Transmit Operation

Isochronous transmits are determined after a valid PCL pointer has been written to the **Packet control list start address register** and the **CH ENA** and **Link** bits have been set as shown in the flow chart in **Error! Reference source not found.**. The overall goal of the isochronous packet processor is to keep the isochronous transmit FIFO full. This means there may be a number of packets queued up in the FIFO especially if the packets are small. Since isochronous packets are "unreliable" in that there is no return status, the DMA will mark the packet complete as soon as it has been transferred to the FIFO.

A transmit operation for isochronous will proceed by checking to see if a wait condition exists. The wait condition is determined by the **Wait Select** bits of the **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18. Since only one packet per channel is allowed during a cycle start period on the bus, the DMA will also optionally wait for a indication from a "cycle start framer". Its job is to watch the isochronous transmit channels and generate a FIFO control word used by the Link Layer Controller to determine the grouping of packets during a cycle start period. This concept is illustrated in **Figure 8: Isochronous Transmit Packet Framing**. The waiting for the cycle start framer can be disabled by **the Multi ISO Packet per Cycle Start** bit in the PCL **data buf0 ctl/byte_cnt/cmd** word**.** The effect of setting the **Multi ISO** bit is global and can affect other ISO transmit channels so all ISO channels should set the Multi ISO bit set to the same value to prevent otherwise unpredicatable behavior.

Figure 8: Isochronous Transmit Packet Framing



When the wait conditions no longer exist, the DMA writes a beginning of packet control word into the transmit DMA and enters a data transfer phase. Here a loop is entered where the current transfer count is checked to see if it has gone to zero. If so, a check is made to see if this is the last data buffer of the PCL buffer list. If there is another buffer in the PCL list then the DMA will acquire the new buffer address and transfer count and proceed with the transfer. While moving data into the asynchronous transmit FIFO from the PCI interface, the DMA will wait for the FIFO to have sufficient room before requesting the PCI bus master to perform a read transfer. Refer to the aforementioned high water mark definition in the receive operation description. While the DMA is transferring data, the **Data buffer start address register** and the **data buffer transfer length** bits in the **DMA control register** are updated to reflect the current state of the transfer.

When the last byte of data from a buffer has been transferred to the isochronous transmit FIFO and the buffer is the last of the PCL list as indicated by the **LAST BUF** bit of the **ctl/byte_cnt** PCL word then the DMA knows that the end of a packet has been reached. The DMA will write a special control word token to the transmit FIFO to mark the end of the packet. The packet count in incremented by 1 to indicate to the Link Layer Controller that a full packet has been loaded into the isochronous transmit FIFO by the DMA. The DMA will update the **DMA status register** with status of 0x0001, the **PKT CMP** is set, and it is then written to the **PCL status** word at PCL offset 0xC along with the number of bytes transferred. If the **INT** bit is set in the **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18 then an interrupt is signaled and latched in the corresponding (**DMA_PCL[x]**) bit in the **Interrupt Status** register.

The DMA then determines whether another PCL has been linked to the current PCL by fetching the **Next List Adr** (PCL offset 0x00). If it is valid as indicated by bit 0 = 0 then the DMA will make this the current PCL address and continue execution as shown. If another PCL had not been linked to the current PCL as indicated by bit 0 = 1 the **Link** and **BSY**

bits are cleared in the **DMA control register,** a DMA halted interrupt is generated for this channel with associated status (**DMA_HLT[x]**) in the **Interrupt Status** register, and the channel becomes idle.

5.2.4.1.4 DMA PCI to LOCAL Bus and LOCAL Bus to PCI Transfers

PCI to/from LOCAL Bus transfer commands transfers data between the PCI bus and the LOCAL Bus. The PCI address and the number of bytes to transfer is derived from the PCL **data buf ctl/byte_cnt/cmd** word(s) in the PCL as for other transfer commands such as transmits. The difference is that the destination or source of the transfer is not the FIFO but rather the LOCAL bus. The LOCAL bus address is generated from the **AUX_ADR** register (see hardware register definitions).

A PCI to/from LOCAL operation will proceed by checking to see if a wait condition exists. The wait condition is determined by the **Wait Select** bits of the **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18. When the wait conditions no longer exist, the DMA enters a loop where the current transfer count is checked to see if it has gone to zero. If so, a check is made to see if this is the last data buffer of the PCL buffer list. If there is another buffer in the PCL list then the DMA will acquire the new buffer address and transfer count and proceed with the transfer. While the DMA is transferring data, the **Data buffer start address register** and the **data buffer transfer length** bits in the **DMA control register** are updated to reflect the current state of the transfer.

When the last byte of data from a buffer has been transferred to/from the LOCAL bus and the buffer is the last of the PCL list as indicated by the **LAST BUF** bit of the **ctl/byte_cnt** PCL word then the DMA knows that the end of the transfer has been reached. The DMA will update the **DMA status register** with status of 0x0001, the **PKT CMP** is set, and it is then written to the **PCL status** word at PCL offset 0xC along with the number of bytes transferred. If the **INT** bit is set in the **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18 then an interrupt is signaled and latched in the corresponding (**DMA_PCL[x]**) bit in the **Interrupt Status** register.

The DMA then determines whether another PCL has been linked to the current PCL by fetching the **Next List Adr** (PCL offset 0x00). If it is valid as indicated by bit 0 = 0 then the DMA will make this the current PCL address and continue execution as shown. If another PCL had not been linked to the current PCL as indicated by bit 0 = 1 the **Link** and **BSY** bits are cleared in the **DMA control register,** a DMA halted interrupt is generated for this channel with associated status (**DMA_HLT[x]**) in the **Interrupt Status** register, and the channel becomes idle.

5.2.4.2 DMA Registers

This function shall implement a control and status register set for controlling and monitoring the status of each DMA channel. The register set shall be implemented for each DMA channel as specified in the register definition section of this specification. The functionality of the register set is defined as follows:

- **Previous Packet control list start address/Temp register**- Updated by the DMA as it processes a queue of packets during asynchronous transmits to keep track of the previous PCL in order to post completion status. It is also used during auxiliary commands as a temporary holding register for load and store data.
- **Packet control list start address register**- Initialized by application software to point to the start of the first (dummy) PCL in a PCL chain. The DMA will use the Next Address loaded in this PCL to link to the first actual PCL. Updated by the active DMA channel as PCLs are processed.
- **Data buffer start address register**- This register is loaded with the data buffer pointers fetched from the PCL as the active DMA channel processes the PCL.
- **DMA status register**- Stores an ongoing count of the number of bytes transferred during this PCL. Contains the completion status of the transfer. After processing of the PCL is completed, the active DMA channel writes the status information of this register back into PCL at offset 0xC.
- **DMA control register**- Contains control bits to allow application software to enable or disable the operation of the DMA channel and re-fetch the next address of a PCL for linkage. Stores the data buffer transfer control, transfer byte count, and commands that are fetched from the PCL .
- **DMA ready register**- The least significant bit of this register can cause the DMA channel to wait for a ready condition before it continues execution of a XMT, RCV, LOAD, STORE, STORE0 or STORE1 command. This ready condition is selected by the control word(s) of the PCL. The least significant bit of this register can cause the DMA channel to conditional branch to during execution of a BRANCH command. This condition is selected by the control word(s) of the PCL.
- **Current DMA state-** Stores the state vector of the DMA channel. This register is updated during the active time of the DMA channel and maintains the last state vector generated when the channel goes inactive.
- **Receive Packet Count Register-** This is a global register that contains the current received packet count. The DMA loads this register with the receive packet count passed in the receive FIFO token words. This count is then decremented as the data is transferred to the PCI bus.
- **DMA Global Register-** This is a global register that contains some state flags used by the state machine to keep track of the execution of an async transmit packet. It also stores the lower bound bits used in conjunction with the cache line size register to determine the burst size requested of the PCI master.

5.2.4.3 DMA Channel Global Issues

There are several global issues dealing with the DMA channel programming.

- Only one DMA channel may be programmed for ASYNC Transmits due to complications with 1394 retries.
- If any ISO transmit channel has the MULTI ISO bit set in the command word of any of its PCL's then all other ISO transmit channels should also have this bit set for proper operation.
- PCL_TO_AUX and AUX_TO_PCL commands use the same AUX address register, so
  - ⇒ Only one channel should be permitted to use these commands and
  - ⇒ PCL's using these commands must reload the AUX addr register as necessary.

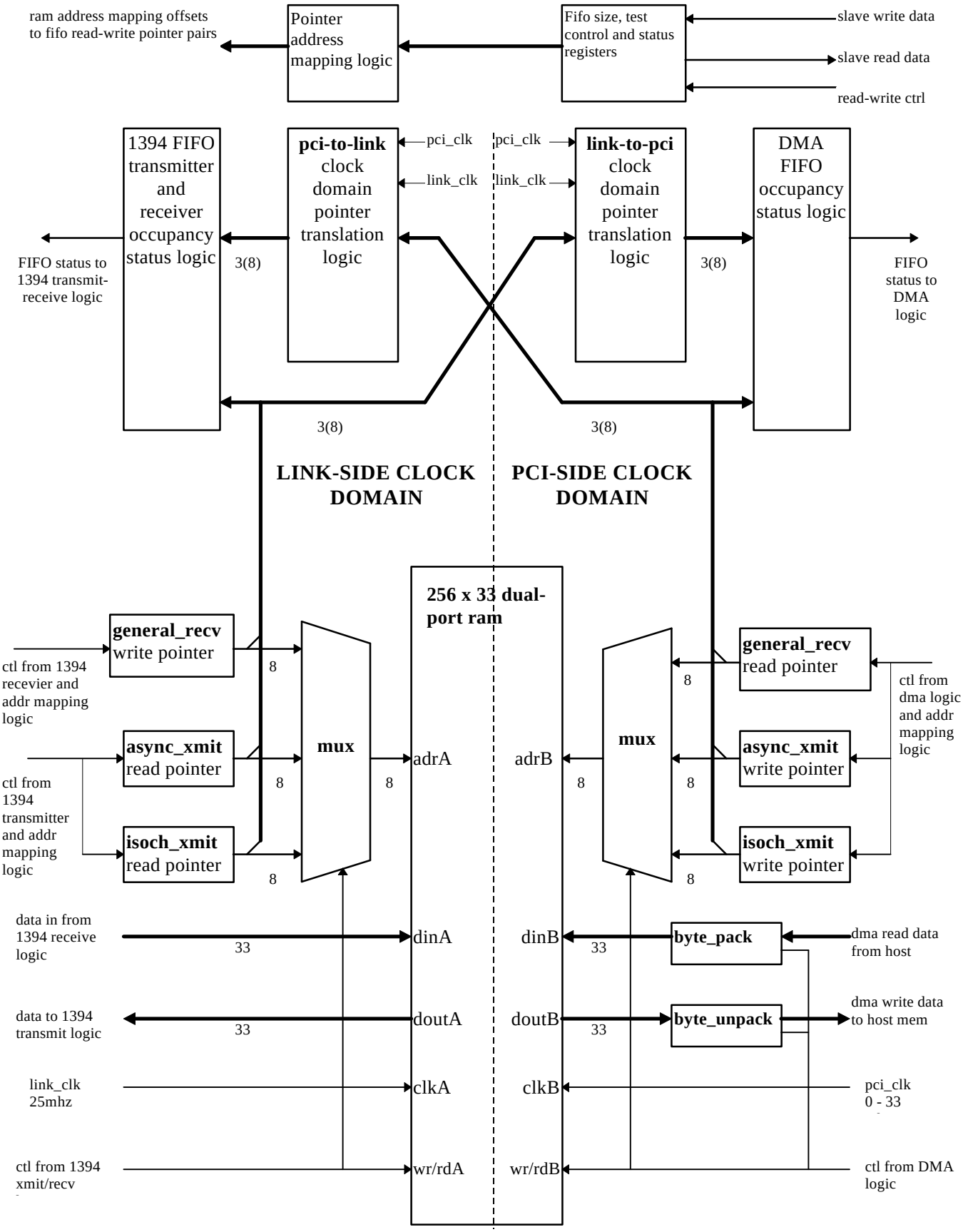
5.2.5 FIFO Logic

This function shall be designed around a single 256 X 33 clocked dual-port ram. The RAM shall be partitioned into three logical FIFO's. Each FIFO shall be programmable in size from 0 to 256 words. For a given combination of FIFO sizes, the sum total of the 3 FIFO sizes shall be less than or equal to 256 words. Each FIFO shall be assigned to a 1394 transfer mode as shown in the table of **Figure 9**. Figure 10 provides a high level functional block diagram of the FIFO.

Figure 9. FIFO Assignments to a 1394 Transfer Mode

FIFO	1394 Receiver	1394 transmitter	active DMA channel
General receive FIFO (GRF)	writes ASYNC or ISOCH packets received from the 1394 bus to the FIFO	NOP	reads the ASYNC or ISOCH packets from the FIFO and writes them to host memory
Asynchronous transmit FIFO (ATF)	NOP	reads ASYNC packets from the FIFO and transmits them over the 1394 bus	reads the ASYNC packets from host memory and writes them into the FIFO
Isochronous transmit FIFO (ITF)	NOP	reads ISO packets from the FIFO and transmits them over the 1394 bus	reads the ISO packets from host memory and writes them into the FIFO

Figure 10. FIFO High level Functional Block Diagram



5.2.5.1 General Receive FIFO

The General Receive FIFO (GRF) shall be comprised of the read and write pointer pair as shown in Figure 10. These pointers shall be used in accessing the dual-port RAM. Each pointer shall count in the range from 0 to it's fifo_size_value minus 1. The ram addressing range for each pointer shall be set by logic which generates an offset value. The offset shall be added to the value of the pointer to map it to a unique range of ram addresses. The read pointer shall be used by the active DMA channel to read asynchronous or isochronous packets from the PCI-side of the RAM, and write them into host memory. The write pointer shall be used by the 1394 receiver to write asynchronous or isochronous packets -received over the 1394 bus- into the link-side of the RAM. The two pointers shall be connected to their respective sides of the RAM thru a multiplexer network.

5.2.5.2 Asynchronous Transmit FIFO

The Asynchronous Transmit FIFO (ATF) shall be comprised of the read and write pointer pair as shown in Figure 10. These pointers shall be used in accessing the dual-port RAM. Each pointer shall count in the range from 0 to it's fifo_size_value minus 1. The ram addressing range for each pointer shall be set by logic which generates an offset value. The offset shall be added to the value of the pointer to map it to a unique range of addresses. The write pointer shall be used by the active DMA channel to write asynchronous packets -that it reads from host memory- into the PCI-side of the RAM. The read pointer shall be used by the 1394 transmitter to read asynchronous packets from the link-side of the RAM, and transmit them over the 1394 bus. The two pointers shall be connected to their respective sides of the RAM thru a multiplexer network.

5.2.5.3 Isochronous Transmit FIFO

The Isochronous Transmit FIFO (ITF) shall be comprised of the read and write pointer pair as shown in Figure 10. These pointers shall be used in accessing the dual-port RAM. Each pointer shall count in the range from 0 to it's fifo_size_value minus 1. The ram addressing range for each pointer shall be set by logic which generates an offset value. The offset shall be added to the value of the pointer to map it to a unique range of addresses. The write pointer shall be used by the active DMA channel to write isochronous packets -that it reads from host memory- into the PCI-side of the RAM. The read pointer shall be used by the 1394 transmitter to read isochronous packets from the link-side of the RAM, and transmit them over the 1394 bus. The two pointers shall be connected to their respective sides of the RAM thru a multiplexer network.

5.2.5.4 FIFO Status Logic

This function shall implement the logic required to generate an occupancy status for each logical FIFO. In computing the PCI-side FIFO status, the link-to-PCI clock domain translation logic shall sample the current value of each pointer on the link side of the fifo and translate these samples from the link clock domain over to the PCI clock domain. Each translated Link-side pointer shall be compared to its corresponding PCI-side pointer to generate an occupancy status for each FIFO. This status shall be used by the DMA logic to pace the transfer of data between host memory and the FIFO. In computing the link-side FIFO status, the PCI-to-link clock domain translation logic shall sample the current value of each pointer on the PCI-side of the FIFO and translate these samples from the pci clock domain over to the link clock domain. Each translated PCI-side pointer shall be compared to its corresponding link-side pointer to compute an occupancy status for each FIFO. This status shall be used by the 1394 transmit-receive logic to pace the transfer of data between the1394 bus and the FIFO.

5.2.5.5 Pointer Dual-Port Address Mapping Logic

This function shall use the three size values from the FIFO size register, to map each of the FIFO read-write pointer pairs, to a unique range of addresses in the dual-port RAM. The pointer address mapping function shall be generated in accordance with the equations as shown in Figure 11 below.

Figure 11. Read-Write Pointer Address Mapping Logic

let ITF = Isochronous Transmit FIFO
let ATF = Asynchronous Transmit FIFO
let GRF = General Receive FIFO

ITF pointer RAM address = ITF _ pointer_ value(0 to (ITF_size - 1)) + 0x00
ATF pointer RAM address = ATF_ pointer _value(0 to (ATF_size - 1)) + ITF_ size
GRF pointer RAM address = GRF _pointer_ value(0 to (GRF_size - 1)) + (ITF_size + ATF_size)

5.2.5.6 Byte Pack Logic

This function shall implement the logic required to assemble a full quadlet using data read from host memory on byte aligned addresses, by the active DMA channel. The logic shall consist of four 8 bit wide registers and four 8-to-1 multiplexers. Each register-mux pair shall correspond to a byte lane. The input of each register shall connect to an input byte lane which is switched by the active DMA channel to host memory. The output of each mux shall connect to an output byte lane, which drives the FIFO. For each 8-to-1 multiplexer, four inputs shall connect in a one-to-correspondence to each register output. The remaining four inputs shall connect in a one-to-one correspondence to each register input. This configuration shall allow byte-aligned DMA read data from the 4 input byte lanes, to be cross-point switched in a different order to the 4 output byte lanes. The control of the byte lane multiplexers shall be performed by the active DMA read channel.

5.2.5.7 Byte Unpack Logic

This function shall implement the logic required to disassemble the quadlet data read from the FIFO, into individually selectable bytes, for writing to host memory on byte aligned addresses by the active DMA channel. This logic shall consist of four 8 bit wide registers and four 8-to-1 multiplexers. Each register-mux pair shall correspond to a byte lane. The input of each register shall connect to an input byte lane, which is driven from the FIFO. The output of each multiplexer shall connect to a output byte lane which is switched by the DMA channel to the host memory. For each of the 8-to-1 multiplexers, four inputs shall connect in a one-to-correspondence to each register output. The remaining four inputs shall connect in a one-to-one correspondence to each register input. This configuration shall allow the quadlet read from the FIFO, to be cross-point switched in a different order onto the output byte lanes. The control of the byte lane multiplexers shall be performed by the active DMA write channel.

5.2.5.8 FIFO Control and Status Registers

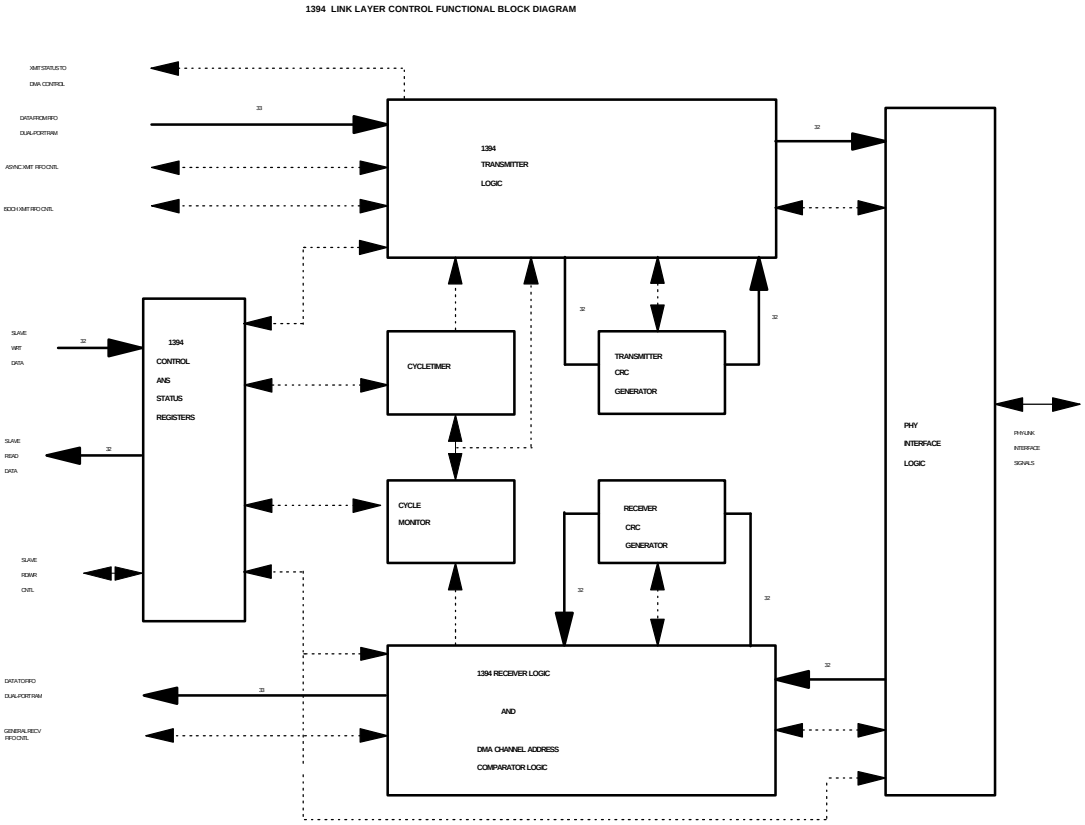
This function shall implement the control and status register set of the FIFO logic. These registers shall be implemented as specified in section 6.4 on page 84 of this specification. The functionality of the register set is summarized as follows:

- **FIFO size register**- used by application software for setting the size of each logical FIFO. This register shall provide three size parameters for programming the size of the ITF, ATF, GRF. This register shall be accessed via a PCI-slave read or write operation.
- **PCI-side FIFO pointer Write-Read port**- shall provide a PCI-slave write-read port for software to fetch the current value of the PCI-side pointers or write a value to them.
- **LINK-side FIFO pointer Write-Read port**- shall provide a PCI-slave read port for software to fetch the current value of the link-side pointers or write a value to them.
- **General Receive FIFO POP-PUSH port**- A 32 bit slave write to this port shall cause the data quadlet to be pushed onto the top of the GRF. A 32 bit slave read from this port shall cause a data quadlet to be popped off the top of the GRF.
- **Asynchronous Transmit FIFO POP-PUSH port**- A 32 bit slave write to this port shall cause the data quadlet to be pushed onto the top of the ATF. A 32 bit slave read from this port shall cause a data quadlet to be popped off the top of the ATF.
- **Isochronous Transmit FIFO POP-PUSH port**- A 32 bit slave write to this port shall cause the data quadlet to be pushed onto the top of the ITF. A 32 bit slave read from this port shall cause a data quadlet to be popped off the top of the ITF.
- **FIFO Control Token Status Read Port**- A slave read from this port shall return the value of bit 33 of the last data quadlet that was popped from one of the three FIFO's that was previously accessed.
- **FIFO Diagnostic test and control register**- shall provide a PCI-slave read-write port for software to configure the FIFO logic for diagnostic testing and control it's operation.
- **Transmit FIFO Threshold register**- Shall provide a PCI-slave read-write port for software to set the Transmit threshold for the ASYN and ISO transmit FIFO's.

5.2.6 1394 Link layer Logic

This function shall implement the 1394 Link Layer Control Logic (LLC) as specified in section 6.0 of the IEEE 1394-1995 standard. This function shall control the transmission and reception of 1394 packet data between the PCI-LYNX FIFO and other devices on the 1394 bus. Figure 12 below provides a high level functional block diagram of the Link Layer Controller logic.

Figure 12. High Level 1394 Link Layer Controller Block Diagram



5.2.6.1 1394 Link Layer Control and Status Registers

This function shall implement the control and status register logic required by application software to control the operation of the LLC and monitor it's operation. This register set shall be implemented as specified in the control and status register definition section of this specification on page 88. The following register set shall be implemented.

- **1394 Bus Number - Node Number register:** Shall provide the interface for application software to program the bus and node numbers.
- **1394 Link Layer Control Register:** Shall provide the interface for application software to control the operating mode of the LLC.
- **1394 Link Layer Interrupt Status Register**: Shall provide the interface for application software to decode the cause of interrupts generated by the LLC and provide a mechanisim for clearing the interrupt status.
- **1394 Link Layer Interrupt Enable register:** Shall provide the interface for application software to selectively enable the status bits in the interrupt status register to generate a LLC interrupt or disable them from generating a LLC interrupt.
- **1394 Cycle Timer Register:** Shall provide the interface for application software to program the cycle timer with an initial value or to read its current value. When the LLC is operating as a cycle master, this timer shall be used to time the transmission of cycle start packets every 125 usec.
- **1394 Physical Layer Access Register:** Shall provide the interface for application software to write data to or read data from the Physical Layer control and status registers
- **1394 Diagnostic test Control:** This register shall provide the interface for application software to perform diagnostic testing of the 1394 LLC logic.
- **DMA Channel 3-0 Word 0 Receive Packet Compare Value Registers:** Their shall be 4 of these registers. Each register shall be assigned to a DMA channel comparator logic function. The DMA channel comparator shall match a selected set of bit positions in the compare value register, to corresponding bit positions of the first quadlet (word 0) of the incoming packet. The bit positions to match shall be specified by the mask value contained in the **Word 0 Receive Packet Compare Mask Register**.
- **DMA Channel 3-0 Word 0 Receive Packet Compare Mask Register:** Their shall be 4 of these registers. Each register shall be assigned to corresponding DMA channel comparator. The DMA channel compare logic shall use the mask value in this register to select the bit positions in word 0 that will be matched against corresponding bit positions in the **Word 0 Receive Compare Value Register.**
- **DMA Channel 3-0 Word 1 Receive Packet Compare Value Registers:** Their shall be 4 of these registers. Each register shall be assigned to a DMA channel comparator logic function. The DMA channel comparator shall match a selected set of bit positions in the compare value register, to corresponding bit positions of the first quadlet (word 1) of the incoming packet. The bit positions to match shall be specified by the mask value contained in the **Word 1 Receive Packet Compare Mask Register**.
- **DMA Channel 3-0 Word 1 Receive Packet Compare Mask Register:** Their shall be 4 of these registers. Each register shall be assigned to corresponding DMA channel comparator. The DMA channel compare logic shall use the mask value in this register to select the bit positions in word 1 that will be matched against corresponding bit positions in the **Word 1 Receive Compare Value Register.**
- **Busy Retry Count Register:** The contents of this register shall specify the number of times the 1394 transmitter should re-try the transmission of an ASYNC packet when a busy acknowledge is received from the destination node. This register shall be read-write by application software via PCI slave access.
- **Busy Retry Transmit Time Interval Register:** The contents of this register shall specify the time interval that the transmitter must delay between successive re-try attempts, when a busy ack is received for each attempt. This register shall be read-write by application software via PCI slave access.
- **State Machine Vector Register**: The register shall provide software with the capability to monitor the state vector of each state machine implemeted in the LLC.
- **FIFO Error Counters**- These counters shall count the under-runs that occur on the ASYNC and ISO transmit FIFO's During packet transmissions and the Over-runs occuring on the GRF during packet reception.

5.2.6.2 1394 Packet Transmit Control Logic

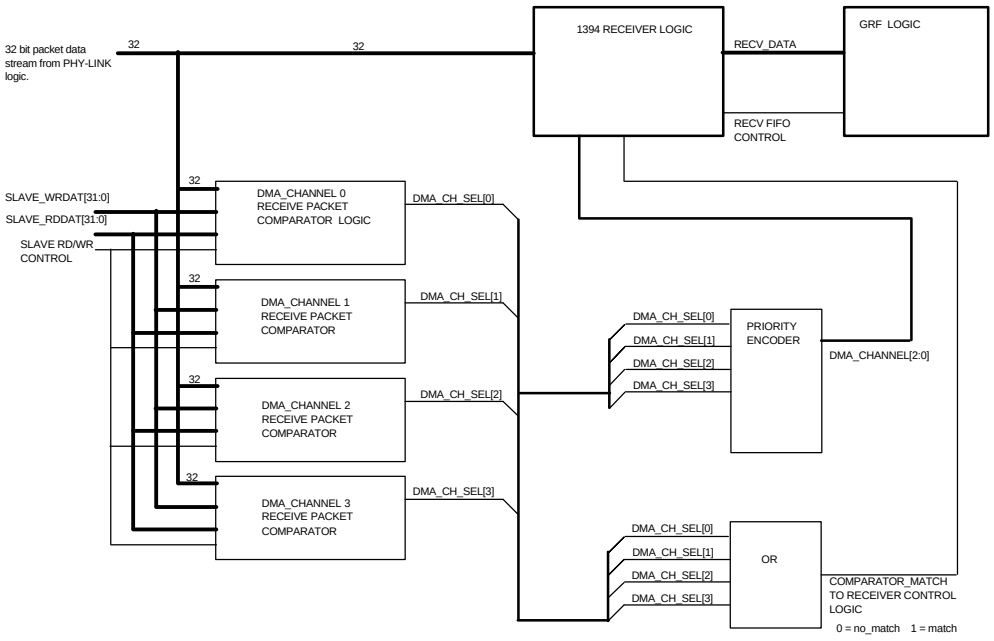
This function shall implement the logic required to control the movement of 1394 packets from either the ITF or ATF to the PHY interface logic for transmission over the 1394 bus. The design of the transmit control logic shall conform to the detail functional requirements as specified in section 6.3 of specification IEEE 1394-1995. The Transmit control logic shall be designed to format the transmit packet formats listed in APPENDIX D - FIFO PACKET ORGANIZATION FORMATS, and the FIFO control token formats listed in APPENDIX E - FIFO CONTROL WORD AND TRANSMIT ACK FORMATS. The following is high-level summary of the functions to be performed.

- Unload quadlets from the asynchronous transmit FIFO and correctly format them into a 32 bit parallel 1394 asynchronous packet stream as specified in section 6.2 of specification IEEE 1394-1995.
- Unload quadlets from the isochronous transmit FIFO and correctly format them into a 32 bit parallel 1394 isochronous packet stream as specified in section 6.2 of specification IEEE 1394-1995.
- Use the CRC logic to compute a CRC code for the header and payload sections of a packet and insert these codes into packet stream in the time slot as required by the format of the packet being transmitted.
- Inputs the parallel packet streams to the PHY-interface logic for conversion from a parallel to a serial data stream format for transmission to the PHY.
- Transmit the cycle start packet when the LLC is programmed to operate as the cycle master.
- Send the 1394 transmit bus requests to the PHY. The PHY layer will arbitrate for the bus and send the indication to the transmitter to start transmitting when the BUS grant is received.
- Execute re-try transmissions using the single phase retryX protocol as specified in IEEE 1394-1995.
- Set the speed of packet transmission

5.2.6.3 DMA Channel Receive Packet Comparator Logic

This function shall implement the logic required to determine if an incoming packet is to be accepted and loaded into the General Receive FIFO. Figure 13 provides a high level functional block diagram of the comparator logic. This function shall implement 4 software programmable comparators. Each comparator shall be assigned to service a DMA channel. A comparator shall be comprised of a word 0 field select register, a word1 field select register, a word 0 compare value register, a word 1 compare value register and the comparison logic. The two field select mask registers shall specify the bit fields in word 0 and word 1 of the incoming packet, that will be matched to an expected value by the comparator logic. The 2 compare value registers shall specify the expected bit patterns that will be matched against the selected bit fields in word 0 and word 1 of the incoming packet. The priority encoder collects the DMA channel match indication from each comparator, and generates a 5 bit code that maps the incoming packet to a DMA channel. The OR gate shall combine the select indications from the four comparators and generate a single comparator match indication to the 1394 receiver logic. The 1394 receiver logic shall use the 5 bit DMA channel number, and comparator match indication to determine if the incoming packet is to be received into the GRF. Refer to Section 6.5 for a detailed definition of the compare value and field select mask registers.

Figure 13: High Level Functional Block Diagram of DMA Channel Receive Packet Comparator Logic



5.2.6.4 1394 CRC Logic

This function shall implement the logic for performing the following functions.

- Generates a 32 bit auto-DIN CRC error code on the header part of the packet data stream generated by the transmitter logic. The transmitter inserts this code into data stream after the header.
- For packets which have a data payload, Generates a 32 bit auto-DIN CRC error code on the data payload portion of the packet stream generated by the transmitter logic. The transmitter inserts this code at the end of the packet stream.
- Generates a 32 bit auto-DIN CRC error code on the header part of an incoming packet data stream. If the computed code is equal to the header CRC code sent with the packet, then the receiver considers the header correct.
- Generates a 32 bit auto-DIN CRC error code on the payload section of an incoming packet data stream. If the computed code is equal to the data CRC code sent with the packet, then the receiver considers the data payload correct.

5.2.6.5 1394 Packet Receiver Control Logic

This function shall implement the logic required to receive incoming 1394 packets. The design of the receiver control logic shall conform to the detail functional requirements as specified in section 6.0 of specification IEEE 1394-1995. The following is high-level summary of the functions to be performed.

- Use the bus and node ID registers and-or the DMA channel Receive Packet Comparators to determine if an incoming asynchronous or isochronous packet is to be accepted.
- Use the CRC logic function to verify correct reception of an incoming packet by checking the header CRC. If the packet has a payload, the data CRC shall be checked.
- Load received packets into the General Receive FIFO if the packet passes the addressing and CRC checks.
- Generate acknowledge on asynchronous receive packets
- Receives cycle start packets.
- Receive self-ID packets and load them into the General Receive FIFO.

5.2.6.6 Cycle Timer Logic

This function shall implement the logic for performing the cycle timer function. The design of this function shall conform to the requirements of a cycle timer function as specified in section 8.0 of the IEEE 1394-1995 standard. The cycle timer shall contain the cycle counter and the cycle offset timer. The offset timer shall either be free running, or reloaded on a low to high transition on the CYCLEIN signal pin, or shall take a reload value from the receiver, based on the state of the CYCLEMASTER and CYCLESOURCE bits in the 1394 LLC control register. This timer shall also be enabled or disabled using the CYCLE_TIMER_ENABLE bit in the 1394 LLC control registers. The Cycle Timer shall be used to support isochronous data transfers. The Cycle Timer shall be 32 bits wide. The low order 12 bits shall count as a modulo 3072 counter, which shall increment once every 24.576 MHz clock period, or (40.69ns). The next 13 high order bits shall be a count of 8khz (or 125usec), and the highest 7 bits shall count in seconds.

5.2.6.7 Cycle Monitor Logic

This function shall implement the logic for performing the cycle monitor function. The cycle monitor shall be used to support isochronous data transfers. It shall monitor the LLC activity and handle the scheduling of isochronous activity. When a cycle start packet is received or transmitted, the cycle monitor shall indicate the occurrence of these events by generating a cycle started or cycle received interrupt. The cycle monitor shall also detect missing cycle start packets and shall generate a cycle lost interrupt. When an isochronous cycle is completed, the cycle monitor shall assert a cycle done interrupt. The cycle monitor shall signal the transmitter to send a cycle start packet when the CYCLEMASTER enable bit is asserted in the 1394 LLC control register.

5.2.6.8 PHY-Link Interface Logic

This function shall implement the logic for interfacing the PCI-LYNX to the physical layer chip. The design of this logic shall conform to the requirements of the LINK-PHY interface specification in annex J of the IEEE 1394-1995 standard. This function shall provide the PCI-LYNX with access to the physical layer services. The following high level functions shall be performed.

- Use the packet speed code from the transmitter, to select the number of serial data streams to generate. If the speed code is set for 100mbps, the parallel data stream is converted into 2 serial data streams each running at 50mbps. If the speed code is set for 200mbps, the parallel data stream is converted into 4 serial data streams each running at 50mbps.
- Use the PHY receive speed indication to convert the incoming serial data streams from the PHY into a parallel data stream for input into the receiver control logic. For any incoming packet, the phy will generate 2 serial data streams to the PCI-LYNX if it is receiving the packet at100 mbps or 4 serial data streams if it is receiving the packet at 200mbps. The serial data streams are each clocked at 50 MHz.
- Detect and receive serial status responses from the PHY and convert them into a parallel format . The status responses shall convey PHY interrupt indications and-or return data in response to a PHY register read access request.
- Detect and receive serial acknowledge packets and convert them into a parallel format
- Accept transmitter packet transmit requests or phy register read-write access requests and format them into a serial request stream for transmission to the PHY.
- Operate with an electrical isolation barrier between the PHY and PCI-LYNX devices.

6. Hardware Register Definitions

6.1 Memory and Configuration Address Space Register Map

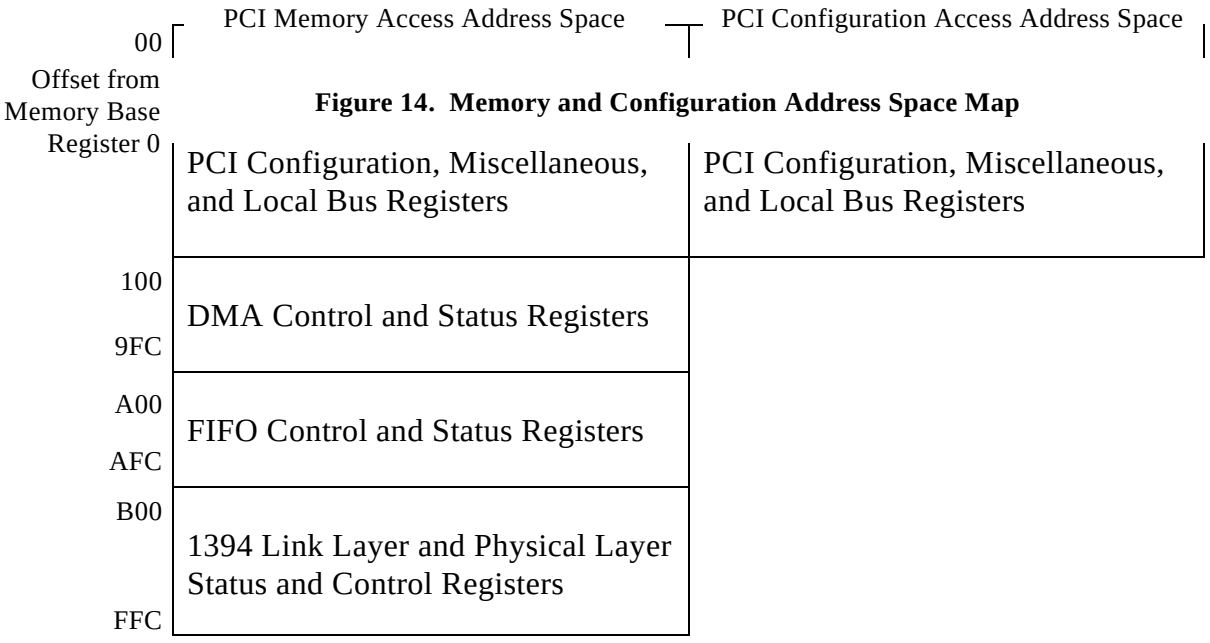


Figure 14. Memory and Configuration Address Space Map

Figure 15. PCI Address Offset Assignments For PCI-LYNX Registers

	PCI Interface and AUX Port Registers			
000	Device ID = 8000		Vendor ID = 0x104C = TI	
004	Status		Command	
008	Class Code = 0x0C0000			Revision ID
00C	BIST	Header Type	Latency Timer	Cache Line Size
010	Memory Base Address Register 0 - Internal PCI-Lynx Registers			
014	Memory Base Address Register 1 - External SRAM on Local Bus			
018	Memory Base Address Register 2 - AUX Port on Local Bus			
01C	Zero			
020	Zero			
024	Zero			
028	Zero			
02C	SubSystem ID		SubSystem Vendor ID	
030	RPL ROM Base Address Register			
034	Zero			
038	Zero			
03C	Max_Latency	Min_Grant	Int_Pin	Int_Line
040	Miscellaneous Control			
044	Serial EEPROM Control Register			
048	PCI Interrupt Status Register			
04C	PCI Interrupt Enable Register			
050	PCI Test Register			
054	Zero			
058	Zero			

0B0	Local Bus Control Register
0B4	Local Bus Address Register
0B8	PCI_GPIO[1:0] Control Register A
0BC	PCI_GPIO[3:2] Control Register B
0C0	PCI_GPIO_DATA_0000 Read-Only Port
0C4	PCI_GPIO_DATA_0001 Read-Write Port
0C8	PCI_GPIO_DATA_0010 Read-Write Port
0CC	PCI_GPIO_DATA_0011 Read-Write Port
0D0	PCI_GPIO_DATA_0100 Read-Write Port
0D4	PCI_GPIO_DATA_0101 Read-Write Port
0D8	PCI_GPIO_DATA_0110 Read-Write Port
0DC	PCI_GPIO_DATA_0111 Read-Write Port
0E0	PCI_GPIO_DATA_1000 Read-Write Port
0E4	PCI_GPIO_DATA_1001 Read-Write Port
0E8	PCI_GPIO_DATA_1010 Read-Write Port
0EC	PCI_GPIO_DATA_1011 Read-Write Port
0F0	PCI_GPIO_DATA_1100 Read-Write Port
0F4	PCI_GPIO_DATA_1101 Read-Write Port
0F8	PCI_GPIO_DATA_1110 Read-Write Port
0FC	PCI_GPIO_DATA_1111 Read-Write Port
	DMA Controller Registers
100	DMA Channel **0** Previous Packet Control List Address/Temp
104	DMA Channel **0** Current Packet Control List Address
108	DMA Channel **0** Current Data Buffer Address
10C	DMA Channel **0** Status
110	DMA Channel **0** Control
114	DMA Channel **0** Ready Register
118	DMA Channel **0** Current State
120	DMA Channel **1** Previous Packet Control List Address/Temp
124	DMA Channel **1** Current Packet Control List Address
128	DMA Channel **1** Current Data Buffer Address
12C	DMA Channel **1** Status
130	DMA Channel **1** Control
134	DMA Channel **1** Ready Register
138	DMA Channel **1** Current State
140	DMA Channel **2** Previous Packet Control List Address/Temp
144	DMA Channel **2** Current Packet Control List Address
148	DMA Channel **2** Current Data Buffer Address
14C	DMA Channel **2** Status
150	DMA Channel **2** Control
154	DMA Channel **2** Ready Register
158	DMA Channel **2** Current State
160	DMA Channel **3** Previous Packet Control List Address/Temp
164	DMA Channel **3** Current Packet Control List Address
168	DMA Channel **3** Current Data Buffer Address
16C	DMA Channel **3** Status
170	DMA Channel **3** Control
174	DMA Channel **3** Ready Register
178	DMA Channel **3** Current State
180	DMA Channel **4** Previous Packet Control List Address/Temp

184	DMA Channel **4** Current Packet Control List Address
188	DMA Channel **4** Current Data Buffer Address
18C	DMA Channel **4** Status
190	DMA Channel **4** Control
194	DMA Channel **4** Ready Register
198	DMA Channel **4** Current State
1A0 to 8E0	This range of address offsets shall be reserved for future use in implementing status and control registers for DMA channels 5 thru 63
900	DMA Diagnostic Test Control Register
904	DMA Receive FIFO Packet Count Registers
908	DMA Global Register
	FIFO Registers
A00	FIFO Size Register
A04	PCI-Side FIFO Pointer Write-Read Port
A08	Link-Side FIFO Pointer Write-Read Port
A0C	FIFO Control Token Status Read-Port
A10	FIFO Control Token Enable and Test Mux Control Register
A14	ATF-ITF Transmit Data Ready Threshold Control Register
A20	General Receive FIFO Pop-Push Port0 FIFO bit32 set to 0 on write
A24	General Receive FIFO Pop-Push Port1 FIFO bit32 set to 1 on write
A28	Not Used
A2C	Not Used
A30	Asynchronous Transmit FIFO Pop-Push Port0 FIFO bit32 set to 0 on write
A34	Asynchronous Transmit FIFO Pop-Push Port1 FIFO bit 32 set to 1 on write
A38	Not Used
A3C	Not Used
A40	Isochronous Transmit FIFO Pop-Push Port0 FIFO bit32 set to 0 on write
A44	Isochronous Transmit FIFO Pop-Push Port1 FIFO bit 32 set to 1 on write
A48 to AFC	Reserved
	Link Layer Controller Registers
B00	DMA channel **0** Word 0 Receive Packet Comparator Value Register
B04	DMA channel **0** Word 0 Receive Packet Comparator Mask Register
B08	DMA channel **0** Word 1 Receive Packet Comparator Value Register
B0C	DMA channel **0** Word 1 Receive Packet Comparator Mask Register
B10	DMA channel **1** Word 0 Receive Packet Comparator Value Register
B14	DMA channel **1** Word 0 Receive Packet Comparator Mask Register
B18	DMA channel **1** Word 1 Receive Packet Comparator Value Register
B1C	DMA channel **1** Word 1 Receive Packet Comparator Mask Register
B20	DMA channel **2** Word 0 Receive Packet Comparator Value Register
B24	DMA channel **2** Word 0 Receive Packet Comparator Mask Register
B28	DMA channel **2** Word 1 Receive Packet Comparator Value Register
B2C	DMA channel **2** Word 1 Receive Packet Comparator Mask Register
B30	DMA channel **3** Word 0 Receive Packet Comparator Value Register
B34	DMA channel **3** Word 0 Receive Packet Comparator Mask Register
B38	DMA channel **3** Word 1 Receive Packet Comparator Value Register

B3C	DMA channel **3** Word 1 Receive Packet Comparator Mask Register
B40	DMA channel **4** Word 0 Receive Packet Comparator Value Register
B44	DMA channel **4** Word 0 Receive Packet Comparator Mask Register
B48	DMA channel **4** Word 1 Receive Packet Comparator Value Register
B4C	DMA channel **4** Word 1 Receive Packet Comparator Mask Register
B50 to EF0	This range of address offsets shall be reserved for future use in implementing comparator control registers for DMA channels 5 thru 63
F00	1394 Bus Number - Node Number Register
F04	1394 Link Layer Control Register
F08	1394 Cycle Timer
F0C	1394 Physical Layer Access Control
F10	1394 Diagnostic Test Control Register
F14	1394 Link Layer Interrupt Status Register
F18	1394 Link Layer Interrupt Enable Register
F1C	1394 Busy Retry Count and retry interval register
F20	LLC State Machine Vector Monitor Port 1
F24	FIFO Overrun-Underrun Error Counters

6.2 PCI Configuration and Miscellaneous Register Definitions

6.2.1 Device-Vendor ID @000

This register provides application software access to the Vendor ID and Device ID numbers that are assigned to the PCI-LYNX ASIC. This register is read-only.

Bit No.	Bit Name	Dir	Description
31 - 16	DEVICE_ID[15:0]	r	Identification number for PCI LYNX = 0x8000 (fixed) Read Only
15 - 00	VENDOR_ID[15:0]	r	Identification Number of Manufacturer = 0x104C = TI (fixed) Read Only

6.2.2 Command - Status @004

This register provides application software with an interface for controlling the PCI-LYNX PCI operating behavior and monitoring the PCI slave and master operation status. This register shall be initialized to 0 on power-up reset. Status bits 31-16 are cleared by writing a “1” to the bit position to be cleared.

Bit No.	Bit Name	Dir	Description
31	PARERR	r/w	PCI-LYNX detected a parity error. parity_detected = 1 no_error = 0
30	SYSERR	r/w	PCI-LYNX asserted SERR# signal. asserted = 1 not_asserted = 0
29	MSTABT	r/w	PCI-LYNX as master aborted PCI transaction. abort = 1
28	TGTABT1	r/w	PCI-LYNX as master was aborted by the target. abort = 1
27	TGTABT0	r/w	PCI-LYNX asserted target abort as a target. abort = 1
26 - 25	DEVSEL[1:0]	r/w	Devsel# timing setting. 01=medium Read Only
24	MSTR_PAR	r/w	PCI-LYNX as master deteced a data parity error or received a PERR signal from the target. data parity error = 1 no data parity error = 0
23	FST0	r	target fast back-to-back capable. not capable = 0 Read Only
22 - 10	reserved	r	Returns 0 when read Read

			Only
09	FST	r	Enable fast back-to-back transaction. disabled = 0 Read Only
08	SEER_ENA	r/w	Enable system error driver. enable = 1 disable = 0
07	WAIT	r	Address/data stepping enable. disabled = 0 Read Only
06	PAR_ENA	r/w	Respond to parity error enable. enable = 1 disable = 0
05	VGA	r	VGA palette snooping enable. disabled = 0 Read Only
04	MWI_ENA	r/w	memory write-invalidate command enable. enable = 1 disable = 0
03	SPC	r	Special cycle operation enable. disabled = 0 Read Only
02	MSTR_ENA	r/w	PCI-LYNX bus master mode enable. enable = 1 disable = 0
01	MEM_ENA	r/w	Memory address space enable. enable = 1 disable = 0
00	I_O_ENA	r	I/O address space access enable. disabled = 0 Read Only

6.2.3 Class Code - Revision ID @008

This register provides an interface for application software to obtain the class and revision code parameters assigned to the PCI-LYNX.

Bit No.	Bit Name	Dir	Description
31 - 08	CLASS_CD[23:0]	r	PCI class code = 0x0C0000 (serial bus / 1394) (fixed) Read Only
07 - 00	REV_ID[7:0]	r	PCI revision code. Set to 0x00 (fixed) Read Only

6.2.4 Header Type- Latency Timer- Cache Line Size @00C

This register provides an interface for application software to program the PCI cache line size, PCI latency timer, PCI header type, and PCI built-in-test parameters. This register shall be set to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31 -24	BIST[7:0]	r	PCI built-in-test(BIST) = 0x00 (no BIST) Read Only
23 - 16	HDR_TYPE[7:0]	r	PCI header type parameter = 0x00 (fixed) Read Only
15 - 08	LAT_TIMER[7:0]	r/w	PCI latency timer parameter
07 - 00	CACHE_LINE_SZ[7:0]	r/w	PCI cache line size parameter

6.2.5 Memory Access Base Address 0 - PCI-Lynx Internal Registers @010

This register provides the interface for application software to set the memory access base address of the internal PCI-Lynx register set. This register shall be set to 0x00000000 on power-up reset (or 0x00010000 on power-up reset if auto_boot is enabled).

Bit No.	Bit Name	Dir	Description
31 - 12	MEMBASE0[31:12]	r/w	Memory access base address register. MEMBASE0[31:12] = PCI LYNX internal register base address
11 - 00	MEMBASE0[11:0]	r	Memory access base address register. MEMBASE0[11:0] = 0x000

6.2.6 Memory Access Base Address 1 - External RAM Port @014

This register provides the interface for application software to set the memory access base address of the external RAM attached to the LYNX Local Bus. This register shall be set to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31 - 16	MEMBASE1[31:16]	r/w	Memory access base address register. MEMBASE1[31:16] = External RAM base address
15 - 00	MEMBASE1[15:0]	r	Memory access base address register. MEMBASE1[15:0] = 0x0000

6.2.7 Memory Access Base Address 2 - AUX Port @018

This register provides the interface for application software to set the memory access base address of the AUX port on the LYNX local bus. This register shall be set to 0x00000000 on power-up reset. The ZV port occupies upper 4K (0xF000 - 0xFFFF) of the AUX port address space when CLK is set to a valid clk in Local Bus Control Register (@0B0).

Bit No.	Bit Name	Dir	Description

31 - 16	MEMBASE2[31:16]	r/w	Memory access base address register. MEMBASE1[31:16] = AUX port base address
15 - 00	MEMBASE2[15:0]	r	Memory access base address register. MEMBASE1[15:0] = 0x0000

6.2.8 SubSystem ID @02C

This register provides a method for the subsystem vendor to uniquely identify his device. The SubSystem Vendor ID is assigned by the PCI SIG to ensure uniqueness. These ID's are loaded from the Serial EEPROM after power reset. This register is set to 0x00000000 on power-up reset

Bit No.	Bit Name	Dir	Description
31 - 16	SubSystem_ID[15:0]	r	unique subsystem ID (initialized from serial EERPOM)
15 - 00	SubSystem Vendor ID[15:0]	r	unique subsystem Vendor ID (initialized from serial EERPOM)

6.2.9 Expansion ROM Base Address @030

This register provides an interface for application software to set the memory access base address of the PCI-LYNX external expansion ROM. To access this address space, both the base address should be set to an appropriate PCI address and the ROM enable bit (bit 0) set to "1". This register shall be set to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31 -16	ROMBASE[31:16]	r/w	PCI-LYNX expansion ROM base address
15 - 01	reserved	r	return 0's for these bits on read
00	ROMEN	r/w	Enable expansion ROM access. enable = 1 disable = 0

6.2.10 Max_Latency - Min_Grant - Int_Pin - Int_Line Register @03C

This register shall be implemented to provide the an interface for application software to read values for the max_latency, min_grant, and interrupt pin parameters, and to write/read values for the interrupt line. The interrupt line register shall be set to 0x00 on power-up reset; the interrupt pin register is read only and fixed at a value of 0x01; the minimum grant and maximum latency registers are loaded from serial EEPROM on power reset. The PCI interface shall return retry status to any accesses while the serial EERPOM machine is active initializing these locations.

Bit No.	Bit Name	Dir	Description
31 - 24	MAX_LAT[7:0]	r	Maximum latency (initialized from serial EERPOM)
23 - 16	MIN_GNT[7:0]	r	Minimum grant time (initialized from serial EERPOM)
15 - 08	INT_PIN[7:0]	r	Interrupt pin used. INTPIN = 0x01 = INTAz
07 - 00	INT_LINE[7:0]	r/w	Interrupt line - indicates which interrupt PCI-LYNX is connected to (set by system software)

6.2.11 Miscellaneous Control @040

This register provides an interface for application software to perform miscellaneous control operations. This register shall also supply operational status information. This register shall be set to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31 - 16	reserved	r	return 0x0000 on a read
15 - 12	MAXRTY_CNT[3:0]	r/w	maximum no. of retries that PCI-LYNX master will attempt when a retry termination status occurs. (0 = infinite number of retries)
11	ENA_MST_RTY	r/w	enable PCI-LYNX master cycle retry count. enable = 1 disable = 0
08 - 10	reserved	r	return 0's on read
07	ENA_POST_WR	r/w	Enable PCI slave posted writes. enable = 1 disable = 0
06	ENA_SLV_BURST	r/w	Enable PCI slave burst. enable = 1 disable = 0
05 - 04	reserved	r	return 0's on read
03	PAUSE_MSTR	r/w	Pause the PCI master on the next access pause = 1 no pause = 0
02	AUTOBOOT_IN	r	Read the value of the autoboot pin. (autoboot = 1)
01	SET_FORCE_INT	w	Set forced interrupt. set = 1. This bit always reads 0.
00	SWRST	w	Software reset. set to 1 to reset. This bit always reads 0.

6.2.12 Serial EEPROM Control @044

This register provides an interface for application software to control the read of the PCI-LYNX external serial EEPROM. This register shall be set to 0x00000000 on power-up reset. Since this register cannot be read until after the internal Serial EEPROM state machine has completed initializing configuration register locations, the value read immediately after power up may not be 0.

The 5 usec timer may be used to time Serial EEPROM accesses. Start the timer by writing a 0 to the timer bit, then poll the register until the timer bit is 1, which will be approximately 5us after starting the timer.

Bit No.	Bit Name	Dir	Description
31 - 10	reserved	r	return zeros on a read
09	EEPERR	r	Serial EEPROM format error
08	EEPCHKERR	r	Serial EEPROM checksum error
07	NOTPRS	r	Serial EEPROM is not present. present = 0 not_present = 1
06	EEPCLK	r/w	Write: Output serial EEPROM clock. high = 1 low = 0 Read: read value of serial EEPROM clock signal
05	EEPENA	r/w	Select serial EEPROM interface output. Serial EEPROM Control reg = 1 PCI-LYNX internal state machine = 0
04	EEPDAT	r/w	Write: Output serial EEPROM data. high = 1 low = 0 Read: read value of serial EEPROM data signal
03	reserved	r	return 0 on read
02	EEPSTARTRD	w	restarts Serial EEPROM read state machine
01	reserved	r	return 0 on read
00	TIMER_5USEC	r/w	5 usec Timer. Time expired = 1 Start timer = 0

6.2.13 PCI Interrupt Status @048

This register provides the interface for application software to determine the events which generate an INTA# interrupt. This register shall be set to 0x00000000 on power-up reset. Interrupt status is cleared by writing a "1" to the bit to be cleared; the interrupt status bit is cleared only if the interrupting condition no longer exists.

Bit No.	Bit Name	Dir	Description
31	INT_PEND	r	Interrupt pending
30	FRC_INT	r/w	forced interrupt set from miscellaneous force interrupt bit
29	Reserved	r	Return 0 on a read.
28	SLV_ADR_PERR	r/w	Slave address parity error
27	SLV_DAT _PERR	r/w	Slave data parity error
26	MST_DAT_PERR	r/w	Master data parity error
25	MST_DEV_TO	r/w	Master Device Timeout
24	MST_RETRY_TO	r/w	Master Retry Timeout
23	INTERNAL_SLV_TO	r/w	Internal slave bus access Timeout
22-19	Reserved	r	Zero Returned on a read for these bits.
18	AUX_TO	r/w	LOCAL BUS Time out
17	AUX_INT	r/w	LOCAL BUS interrupt
16	P1394_INT	r/w	1394 interrupt from Link Layer
10-15	reserved	r	Returns 0 when read
09	DMA4_PCL	r/w	DMA channel 4 Packet Control List caused interrupt. Interrupt = 1. Writing a 1 to this bit clears this interrupt.
08	DMA4_HLT	r/w	DMA 4 channel was halted. Interrupt = 1. Writing a 1 to this bit clears this interrupt
07	DMA3_PCL	r/w	DMA channel 3 Packet Control List caused interrupt. Interrupt = 1. Writing a 1 to this bit clears this interrupt.
06	DMA3_HLT	r/w	DMA 3 channel was halted. Interrupt = 1. Writing a 1 to this bit clears this interrupt
05	DMA2_PCL	r/w	DMA channel 2 Packet Control List caused interrupt. Interrupt = 1. Writing a 1 to this bit clears this interrupt.
04	DMA2_HLT	r/w	DMA 2 channel was halted. Interrupt = 1. Writing a 1 to this bit clears this interrupt
03	DMA1_PCL	r/w	DMA channel 1 Packet Control List caused interrupt. Interrupt = 1. Writing a 1 to this bit clears this interrupt.
02	DMA1_HLT	r/w	DMA 1 channel was halted. Interrupt = 1. Writing a 1 to this bit clears this interrupt
01	DMA0_PCL	r/w	DMA channel 0 Packet Control List caused interrupt. Interrupt = 1. Writing a 1 to this bit clears this interrupt.
00	DMA0_HLT	r/w	DMA 0 channel was halted. Interrupt = 1. Writing a 1 to this bit clears this interrupt

6.2.14 PCI Interrupt Enable @04C

This register provides an interface for application software to selectively enable the events which can cause an interrupt to occur. This register shall be set to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31	INT_PEND	r	Interrupt pending
30	FRC_INT_EN	r/w	Enable forced interrupt
29	reserved	r	Returns 0 when read.
28	SLV_ADR_PERR_EN	r/w	Enable Slave address parity error interrupt

27	SLV_DAT _PERR_EN	r/w	Enable Slave data parity error interrupt
26	MST_DAT_PERR_EN	r/w	Enable Master data parity error interrupt
25	MST_DEV_TO_EN	r/w	Enable Master Device Timeout interrupt
24	MST_RETRY_TO_EN	r/w	Enable Master Retry Timeout interrupt
23	INT_SLV_TO_EN	r/w	Enable Internal Slave Timeout interrupt
22-19	reserved	r	Enable Zero Returned on a read for these bits.
18	AUX_TO_EN	r/w	Enable LOCAL BUS Time out interrupt
17	AUX_INT_EN	r/w	Enable LOCAL BUS interrupt
16	P1394_INT_EN	r/w	Enable Link Layer interrupt. Input fed by LINK_INT in 1394 Interrupt Status Register, masked by LLC_INT_EN in 1394 Interrupt Enable Reg.
15-13	reserved	r	Returns 0 when read
09	DMA4_PCL_EN	r/w	DMA ch4 Packet Control List interrupt enable. enable = 1 disable = 0
08	DMA4_HLT_EN	r/w	DMA ch4 halted interrupt enable. enable = 1 disable = 0
07	DMA3_PCL_EN	r/w	DMA ch3 Packet Control List interrupt enable. enable = 1 disable = 0
06	DMA3_HLT_EN	r/w	DMA ch3 halted interrupt enable. enable = 1 disable = 0
05	DMA2_PCL_EN	r/w	DMA ch2 Packet Control List interrupt enable. enable = 1 disable = 0
04	DMA2_HLT_EN	r/w	DMA ch2 halted interrupt enable. enable = 1 disable = 0
03	DMA1_PCL_EN	r/w	DMA ch1 Packet Control List interrupt enable. enable = 1 disable = 0
02	DMA1_HLT_EN	r/w	DMA ch1 halted interrupt enable. enable = 1 disable = 0
01	DMA0_PCL_EN	r/w	DMA ch0 Packet Control List interrupt enable. enable = 1 disable = 0
00	DMA0_HLT_EN	r/w	DMA ch0 halted interrupt enable. enable = 1 disable = 0

6.2.15 PCI Test Register @050

This register provides the interface for application software to enable various test modes and functions in the LYNX. This register shall be set to 0x00000000 on power-up reset. Normal application software should not write this register. TEST_REG_EN must be set before SET_OUTPUT_FF or DISABLE_DRIVERS are functional.

Bit No.	Bit Name	Dir	Description
31-14	reserved	r	Returns 0 when read
13	TEST_OUTPUT	r	Test output from test mux.
12-08	TEST_MUX_SEL	r/w	Test mux select.
07-04	reserved	r	Returns 0 when read
02	SET_OUTPUT_FF	r/w	Set output flip-flops. set = 1 normal = 0
01	DISABLE_DRIVERS	r/w	Tri-states normally output-only drivers. Tri-stated = 1, normal = 0
00	TEST_REG_EN	r/w	Test register enable. enable = 1 disable = 0

TEST_MUX_SELECT [4:0] (hex)	Selected Signal
0	test_nand_chain output
1	dma_test_mux output
2	fifo_test_mux output
3	link_test_mux output
4	PCI module - mstr_act
5	PCI module - mstr_err
6	PCI module - mstr_req
7	PCI module - mstr_xfr
8	PCI module - mstr_ack
9	PCI module - mstr_internal_cyc
A	PCI module - my_slv_cyc
B	PCI module - slv_data
C	PCI module - slv_rd
D	PCI module - slv_wr
E	PCI module - int_slv_xfr
F	PCI module - int_pci_slv_act
10	PCI module - internal_mstr_act
11	PCI module - pci_mstr_st
12	PCI module - aux_pci_act
13	PCI module - m_addr
14	PCI module - m_data
15	PCI module - pci_mstr_xfrq
16	PCI module - mstr_byte_cnt_eq_0
17	PCI module - pci_xfr_cnt_eq_0
18	PCI module - in_buf_full
19	PCI module - wr_buf_full
1A	PCI module - wr_buf_empty
1B	PCI module - mstr_fifo_rdy
1C	PCI module - aux_busy
1D	PCI module - zv_busy
1E	PCI module - retry_slv
1F	PCI module - reserved

6.2.16 Local Bus Control Register @0B0 { ROM, RAM, AUX, and ZV registers }

This register provides an interface for application software to control the pacing of data transferred on the Local Bus to/from attached external devices. This register shall also specifies the data bus width required for each external device type and the polarity of incoming interrupts. This register shall be initialized to 0x00000000 on power up reset. Each byte controls a different area - ROM, RAM, AUX and ZV.

Bit No.	Bit Name	Dir	Description
31	GATE_PIXEL_CLK	r/w	ZV pixel clock gateing enable, 1=gateing enabled, 0=free running pixel clk
30 - 28	HSYNC_CNT[2:0]	r/w	Horizontal sync count 0 0 0 Cirrus mode (target contains internal hsync counter) Hsync will still be generated for every frame. 0 0 1 hsync cnt = 1 . . 1 1 1 hsync cnt = 15
27 - 25	ZV_CLK[2:0]	r/w	ZV pixel clock select, Note: **ONLY** those denoted with a "*" are valid selections for 8-bit mode. AUX address space F000 - FFFF is allocated to ZV (ZV is enabled) when one of the 6 available ZV pixel clock sources are selected. 0 0 X ZV port disabled 0 1 0 external clock 0 1 1 external clock / 2***** 1 0 0 sclk / 2 (25.0 MHz) 1 0 1 sclk / 4 (12.5 Mhz)***** 1 1 0 pci_clk 1 1 1 pci_clk / 2*****
24	ZV_16/8	r/w	Data Width, 1= ZV access is 16 bits wide, 0= ZV access is 8 bits wide
23 - 20	AUX_WS[3:0]	r/w	Number of wait states to insert for External AUX access
19	Reserved	r	Zero returned on read
18	AUX_INT_POL	r/w	AUX interrupt polarity, 1= invert, 0=don't invert
17	AUX_RST	r/w	AUX port reset output
16	AUX_16/8	r/w	Data Width, 1= AUX access is 16 bits wide, 0= AUX access is 8 bits wide
15 - 12	RAM_WS[3:0]	r/w	Number of wait states to insert for External RAM access
11 - 09	Reserved	r	Zeros returned on read
08	RAM_16/8	r/w	Data Width, 1= RAM access is 16 bits wide, 0= RAM access is 8 bits wide
07 - 04	ROM_WS[3:0]	r/w	Number of wait states to insert for External ROM access
03 - 02	Reserved	r	Zeros returned on read
01	ROM_WR_EN	r/w	ROM Write Enable (writable non-volatile memory)
00	ROM_16/8	r/w	Data Width, 1= ROM access is 16 bits wide, 0= ROM access is 8 bits wide

6.2.17 Local Bus Address Register @0B4

This port provides application software with an interface to specify the Local Bus address to be used for DMA transfers from the Local Bus to the PCI Bus. This address auto increments every time it is used. This address must be specifically written to reinitialize. This register shall be initialized to 0x00000000 on power up reset.

Bit No.	Bit Name	Dir	Description
31 - 18	Reserved	r	Zeros returned on read
17 - 02	AUX_ADR[17:02]	r/w	AUX address to use during slave reads and writes. Address autoincrements every time it is used and the high byte enable (3) is valid. Must be re-written to reinitialize.
01 - 00	Reserved	r	Zeros returned on read

6.2.18 PCI_GPIO[1:0] Control Register A @0B8

This register provides application software with an interface for configuring the operating mode of GPIO[1:0] port pins. This register shall be initialized to 0x00000000 on power up reset. Normally each 16 bit register half is accessed separately.

Bit No.	Bit Name	Dir	Description
31 - 29	reserved	r	Zeros returned on read
28 - 24	GPIO_SRC1[4:0]	r/w	Data bit mux select for output on GPIO[1]
23 - 19	reserved	r	Zeros returned on read
18	GPIO_POL_OUT1	r/w	GPIO[1] output polarity control (0 = non-inverted 1 = inverted)
17	reserved	r	Zero returned on read
16	GPIO_OUT_EN1	r/w	GPIO[1] output enable control (0 = tri-state; 1 = enabled)
15 - 13	reserved	r	Zeros returned on read
12 - 08	GPIO_SRC0[4:0]	r/w	Data bit mux select for output on GPIO[0]
07 - 03	reserved	r	Zeros returned on read
02	GPIO_POL_OUT0	r/w	GPIO[0] output polarity control (0 = non-inverted 1 = inverted)
01	reserved	r	Zero returned on read
00	GPIO_OUT_EN0	r/w	GPIO[0] output enable control (0 = tri-state; 1 = enabled)

6.2.19 PCI_GPIO[3:2] Control Register B @0BC

This register provides application software with an interface for configuring the operating mode of GPIO[3:2] port pins. This register shall be initialized to 0x00000000 on power up reset. Normally each 16 bit register half is accessed separately.

Bit No.	Bit Name	Dir	Description
31 - 29	reserved	r	Zeros returned on read
28 - 24	GPIO_SRC3[4:0]	r/w	Data bit mux select for output on GPIO[3]
23 - 19	reserved	r	Zeros returned on read
18	GPIO_POL_OUT3	r/w	GPIO[3] output polarity control (0 = non-inverted 1 = inverted)
17	GPIO_RDY_POL3	r/w	GPIO[3] input polarity control (0 = non-inverted 1 = inverted)
16	GPIO_OUT_EN3	r/w	GPIO[3] output enable control (0 = tri-state; 1 = enabled)
15 - 13	reserved	r	Zeros returned on read
12 - 08	GPIO_SRC2[4:0]	r/w	Data bit mux select for output on GPIO[2]
07 - 03	reserved	r	Zeros returned on read
02	GPIO_POL_OUT2	r/w	GPIO[2] output polarity control (0 = non-inverted 1 = inverted)
01	GPIO_RDY_POL2	r/w	GPIO[2] input polarity control (0 = non-inverted 1 = inverted)

00	GPIO_OUT_EN2	r/w	GPIO[2] output enable control (0 = tri-state; 1 = enabled)

6.2.20 PCI GPIO DATA Read-Write Ports @0C0 thru @0FC

The PCI address offsets indicated in the following table, shall be used by the application software to perform PCI read or writes from or to various combinations of GPIO ports.

The PCI address offset written to, shall determine exactly the combination of GPIO ports that are actually written. PCI address bit 2 enables GPIO[0] writes, bit 3 enables GPIO[1] writes, bit 4 enables GPIO[2] writes, and address bit 5 enables GPIO[3] writes.

PCI slave writes to these address offsets must write to at least byte 0 to write to any GPIO port.

The data bit value that is written to a GPIO port shall be selected from the 32 bit PCI slave write data value using a 32:1 data bit mux. There are four of these muxes, one for each GPIO port. The bit select control for each of the muxes, shall be set by the value of the GPIO_SRCx[4:0] mux select field. These fields are specified in the GPIO[1:0] and GPIO[3:2] control registers.

A read of the GPIO register always returns the value read from all four GPIO ports.

PCI Address Offset	GPIO Ports Written to	GPIO Ports Read
0C0	None - NOP	GPIO[3:0]
0C4	GPIO[0]	GPIO[3:0]
0C8	GPIO[1]	GPIO[3:0]
0CC	GPIO[1,0]	GPIO[3:0]
0D0	GPIO[2]	GPIO[3:0]
0D4	GPIO[2,0]	GPIO[3:0]
0D8	GPIO[2,1]	GPIO[3:0]
0DC	GPIO[2,1,0]	GPIO[3:0]
0E0	GPIO[3]	GPIO[3:0]
0E4	GPIO[3,0]	GPIO[3:0]
0E8	GPIO[3,1]	GPIO[3:0]
0EC	GPIO[3,1,0]	GPIO[3:0]
0F0	GPIO[3,2]	GPIO[3:0]
0F4	GPIO[3,2,0]	GPIO[3:0]
0F8	GPIO[3,2,1]	GPIO[3:0]
0FC	GPIO[3,2,1,0]	GPIO[3:0]

6.3 DMA Control and Status Register Definitions

6.3.1 DMA channel 0 thru 4 - Previous packet Control List Address/Temp @100 120 140 160 180

This register contains the address of the previous packet control list which is being processed by the active DMA channel when programmed for asynchronous transmit operations. This register is also used during the execution of auxiliary commands to temporarily hold data during a load or store command. This register shall be initialized to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31 - 00	PPLADR[31:0]	r/w	Previous Packet control list address or temporary data during auxiliary store or load commands.

6.3.2 DMA channel 0 thru 4 - Current packet Control List Address @104 124 144 164 184

This register shall specify the address of the current packet control list which is being processed by the active DMA channel. This register shall be initialized by application software to point to a PCL with a valid next address pointing to the start of the first packet control list which is the first in a queue of control lists to be processed. The active DMA channel shall update this register with start of a packet control list as it steps thru the queue. This register shall be initialized to 0x00000001 on power-up reset in non-autoboot mode. This register shall be initialized to 0x00000000 on power-up reset in autoboot mode.

Bit No.	Bit Name	Dir	Description
31 - 04	CPLADR[31:4]	r/w	Packet control list address - High order address bits.
03 - 01	CPLADR[3:1]	r/w	Packet control list boundary - set = 000 to be on cache line boundary
00	CPLVALID	r/w	Packet control list address not valid. not_valid = 1 valid = 0

6.3.3 DMA channel 0 thru 4 - Current Data Buffer Address @108 128 148 168 188

This register shall contain the start address of the host memory data buffer that is being processed by the active DMA channel. The active DMA channel loads this register with the data buffer start address that is obtained from the current packet control list. This register is used by the DMA to read in the next PCL address for validation. As a result it may change during a PCL link operation. This register shall be initialized to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31 - 00	CDBADR[31:0]	r/w	data buffer start address currently being processed by the active DMA channel.

6.3.4 DMA channel 0 thru 4 - DMA channel status @10C 12C 14C 16C 18C

This register contains ongoing status and byte count logging during the execution of a PCL. The active DMA channel shall store status from this register back at packet control list offset 0xC. This register shall be initialized to 0x000000000 on power-up reset. This is roughly the same information as appears in in the PCL's Status. The PCL should be consulted instead if DMA is still running since this register can be changing as the DMA processes multiple PCL's.

<table>
<tr><th colspan="4">Status</th></tr>
<tr><td>Bit No.</td><td>Bit Name</td><td colspan="2">Description</td></tr>
<tr><td>31</td><td>reserved</td><td colspan="2">Written with unknown data by the DMA.</td></tr>
<tr><td>30</td><td>ISO MODE</td><td colspan="2">The Received Packet was an ISO Packet.</td></tr>
<tr><td>29</td><td>Mst Err</td><td colspan="2">PCI Master Error. Set to a 1 by the DMA if it receives an error indication (parity error, timeout, etc.) from the PCI Master during execution of this PCL. In general this is a fatal condition which will cause the channel to stop, the LINK, BSY, and ENA bit are cleared in the DMA Command register (see register definitions) and an DMA_HLT interrupt (see Interrupt Status Register)
will be generated if enabled. Cleared by writing a 1 by software. Also writeable with the opposite of write data when DMATESTEN is 1.</td></tr>
<tr><td>28</td><td>Pkt Err</td><td colspan="2">Packet Error. Set to a 1 by the DMA for any transfer to or from the 1394 bus in which the transfer had an error. The error can be determined from the Ack_Type and Acks fields. Pkt Err may not be set if Mst Err is set since it may be impossible for the DMA to update the PCL.</td></tr>
<tr><td>27</td><td>Pkt Cmp</td><td colspan="2">Packet Complete. Written by the DMA upon completion of this packet.</td></tr>
<tr><td rowspan="8">26-21</td><td rowspan="8">Receive Dma_Cha[5:0]</td><td colspan="2">Received DMA Channel number. This is the Channel number received from the Link Controller via the receive FIFO control word. Valid only for channels programmed for receive operations. These bits shall return zeros for other commands.</td></tr>
<tr><td>Dma_Cha[5:0]</td><td>DMA Channel Number</td></tr>
<tr><td>0 0 0 0 0 0</td><td>0</td></tr>
<tr><td>0 0 0 0 0 1</td><td>1</td></tr>
<tr><td>0 0 0 0 1 0</td><td>2</td></tr>
<tr><td>0 0 0 0 1 1</td><td>3</td></tr>
<tr><td>0 0 0 1 0 0</td><td>4</td></tr>
<tr><td>Others</td><td>reserved</td></tr>
<tr><td>20 - 19</td><td>Rcv_Speed[1:0]</td><td colspan="2">The speed at which the packet was received for Asynchronous or Isochronous Transfers. Valid only for channels programmed for receive operations. These bits shall return zeros for other commands.
00 = 100 Mbps.
01 = 200 Mbps</td></tr>
</table>

18 - 15	Acks	Packet Acknowledge. Ack status returned from the Link Layer Controller for this packet. Written by the DMA upon completion of this packet. These bits are written with zeros after completion of Auxiliary commands. These bits are written with 0x0001 after completion of an isochronous transmit or PCI to/from local bus transfers. These bit also contain a special code for internally (non 1394) related errors when bit 14 (Ack_Type) is set. The encoding for these errors are as follows: 0000 = Link reported a Retry Overrun 0001 = Link reported a Timeout 0010 = Link reported a FIFO underrun 0101 = No expected End of receive Packet 0110 = Pipelined Async Transmit Command encountered a command other than another Async Transmit. 1110 = Link reported a corrupted header before the packet was transmitted.
14	Ack_Type	Acknowledge type returned by 1394 Transmitter logic Ack_Type = 0 indicates a normal 1394 ack code is returned in bits 18 - 15 Ack_Type = 1 indicates a special ack code is returned in bits 18 - 15
13	reserved	Written with unknown data by the DMA.
12-00	Transferred Count	For all RCV and isochronous XMT commands, the DMA will update these bits with the total number of bytes transferred for this packet. These bits are indeterminate for asynchronous transmits due to the potentially pipelined nature of asynchronous XMT commands. These bits are written with zeros after completion of auxiliary commands.

6.3.5 DMA channel 0 thru 4 - DMA channel control @110 130 150 170 190

This register shall provide the interface for application software to initiate the operation a DMA channel and to monitor its operational status. This register shall be initialized to 0x000000000 on power-up reset in non-autoboot mode or 0xa0000000 (CH ENA and LINK) in autoboot mode. This is roughly the same information as appears in in the PCL's Control. The PCL should be consulted instead if DMA is still running since this register can be changing as the DMA processes multiple PCL's.

Bit No.	Bit Name	Dir	Description
31	CH ENA	r/w	Write 1: Starts the DMA packet processing engine. Write 0: DMA packet processing engine will stop at the end of the current packet and go idle.
30	BSY	r	1= DMA packet processing engine is currently processing a PCL queue. 0= DMA packet processing engine is idle waiting for a valid PCL to process.
29	LINK	r/w	1= DMA is to fetch or re-fetch the Next Address entry of the current PCL and perform a check on the valid bit. If the valid bit is set, the DMA will make the Next Address entry the Current PCL and continue execution. 0= The DMA clears this bit when it encounters an invalid Next Address or Next Stream Address entry in the Current PCL. The DMA will stop at this point and wait for a valid Current PCL address, LINK set to a 1, and a valid Next Address entry in the Current PCL.
28	reserved	r	These bit shall be ignored when read and written with 0.
27-24	CMD3-0	r	Command Select. These bits control what command the DMA channel will execute. Loaded by the DMA from PCL offset 0x18.

CMD[3:0]	Command
0000	NOP. (NOP parameter fetched but ignored)
0001	RCV. (1394 FIFO to memory)
0010	XMT. (Memory to 1394 FIFO)
0011	LOAD. (@DESTINATION => TEMP)
0100	STORE_QUAD (4 bytes TEMP => @SOURCE)
0101	STORE0. (00000000 => @DESTINATION)
0110	STORE1. (FFFFFFFF => @DESTINATION)
0111	Conditional BRANCH to DESTINATION if the conditions are met as specified in the condition field. Status is updated and an interrupt is generated, if enabled, prior to the branch.
1000	PCI_TO_LBUS.
1001	LBUS_TO_PCI.
1010	RCV_AND_UPDATE.
1011	STORE _DOUBLE. (2 bytes TEMP => @SOURCE)
1100	UNFORMATTED_XMT.
1101	ADD.
1110	COMPARE.
1111	reserved

23	reserved	r	These bit shall be ignored when read and written with 0.

<table>
<tr><td>22-20</td><td>Condition Codes 2-0
(branch command)</td><td>r</td><td>Branch Command Condition codes. These bits select what conditions have to be met during the execution of the BRANCH command to cause the address contained in DESTINATION to be loaded into the NEXT PCL ADDRESS and linked. Loaded by the DMA from PCL offset 0x18. (..0x24, 0x30...0x78)<table><tr><td>Condition Code [2:0]</td><td>Branch Condition</td></tr><tr><td>000</td><td>Don’t Branch</td></tr><tr><td>001</td><td>Branch if DMA Ready Register = 1 (this chan)</td></tr><tr><td>010</td><td>Branch if DMA Ready Register = 0 (this chan)</td></tr><tr><td>011</td><td>Branch if External Ready pin RDY = 1 (this chan)</td></tr><tr><td>100</td><td>Branch if External Ready pin RDY = 0 (this chan)</td></tr><tr><td>101</td><td>Branch if GPIO port 2 is active (this chan)</td></tr><tr><td>110</td><td>Branch if GPIO port 3 is active (this chan)</td></tr><tr><td>111</td><td>Reserved</td></tr></table></td></tr>
<tr><td>22-20</td><td>Wait Sel 2-0
(all commands except branch)</td><td>r</td><td>Wait Select. These bits control what conditions have to be met before execution of the PCL will continue. Loaded by the DMA from PCL offset 0x18.<table><tr><td>Wait Select [2:0]</td><td>Wait Condition</td></tr><tr><td>000</td><td>Don’t Wait, Continue execution.</td></tr><tr><td>001</td><td>Wait for DMA Ready Register = 1 (this chan)</td></tr><tr><td>010</td><td>Wait for DMA Ready Register = 0 (this chan)</td></tr><tr><td>011</td><td>Wait for External Ready pin RDY = 1 (this chan)</td></tr><tr><td>100</td><td>Wait for External Ready pin RDY = 0 (this chan)</td></tr><tr><td>101</td><td>Wait for GPIO port 2 to go active (this chan)</td></tr><tr><td>110</td><td>Wait for GPIO port 3 to go active (this chan)</td></tr><tr><td>111</td><td>Reserved</td></tr></table></td></tr>
<tr><td>19</td><td>INT</td><td>r</td><td>Generate interrupt to host upon completion of packet control list. An interrupt will be generated by the DMA regardless of the state of this bit in the case of an error resulting in Pkt Err or Mst Err status being set.
interrupt enabled = 1 disabled = 0 Loaded by the DMA from PCL offset 0x18.</td></tr>
<tr><td>18</td><td>LAST BUF</td><td>r</td><td>Last Buffer indicator. Indicates the end of a packet.
Loaded by the DMA from the PCL.</td></tr>
<tr><td>17</td><td>WAIT FOR STATUS</td><td>r</td><td>Is used to ‘single thread’ asynchronous transmits. Normally, transmits of asynchronous transmits are pipelined to improve throughput. Setting this bit will cause the DMA to wait for transmit completion status before continuing.
Loaded by the DMA from PCL offset 0x18.</td></tr>
<tr><td>16</td><td>BIG ENDIAN</td><td>r</td><td>Byte ordering. Controls the byte ordering of the data buffer as it is read or written.
NOTE: The Big Endian flag may only be changed on quadlet boundaries. I.E. between header and payload data.
0=Little Endian (3,2,1,0)
1=Big Endian (0,1,2,3)
Loaded by the DMA from PCL offset 0x18.</td></tr>
<tr><td>15-14</td><td>xmt_spd_code[1:0]</td><td>r</td><td>1394 transmit speed code. Specifies the transmission speed of an asynchronous or isochronous transmit packet. xmt_spd_code[1:0] = 00 - 100mbps
xmt_spd_code[1:0] = 01 - 200mbps
The value of this field is only valid for DMA transmit commands.</td></tr>
</table>

13	Multi ISO Packet per Cycle Start	r	This bit is relevant for an isochronous DMA channel (ISO Mode = 1). 0=This isochronous packet should be sent with regard to cycle start boundaries. One isochronous packet per isochronous DMA channel per cycle start period. 1=This isochronous packet should be sent without regard to cycle start boundaries. This implies multiple isochronous packets for the same DMA channel may be transmitted during a cycle start period. The effect of setting this bit is global and can affect other ISO transmit channels so all ISO channels should set the Multi ISO bit set to the same value to prevent otherwise unpredicatable behaviour. Loaded by the DMA from PCL offset 0x18.
12	Transmit ISO Mode	r	0= This DMA channel is to be configured for transmit asynchronous transfers. 1=This DMA channel is to be configured for transmit isochronous transfers. Loaded by the DMA from PCL offset 0x18. Also must be written (1 or 0) whenever this channel's LINK bit is set. This is required to insure proper fairness of ISO XMT's if more than one DMA channel is to be used for ISO XMT's.
11 - 00	DBXXFRLEN[11:0]	r	Remaining count to be transferred of the current buffer. Loaded by the DMA from the PCL.

6.3.6 DMA channel 0 thru 4 - DMA Ready Register @114 134 154 174 194

This register shall be implemented to provide a mechanism for pacing the DMA. The wait select bits in the PCL **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18 can select the contents of this register to be used in a decision to halt this channel until the wait condition no longer exists. Writes to this register by software or by another channel's auxiliary store commands can modify the wait condition.
The auxiliary branch command can also use this register to conditionally branch. The condition select bits in the PCL **data buf0 ctl/byte_cnt/cmd** at PCL offset 0x18 can select the contents of register to be used in a decision to branch to another PCL. This register is set to 0x000000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31-01	unused	r/w	return 0 when read, ignored when written
00	CONDITION	r/w	This bit is used for wait or branch conditional checks.

6.3.7 DMA channel 0 thru 4 - Current DMA state @118 138 158 178 198

This register shall be implemented to provide an interface for application software to read the state_vector of a DMA channel. The active DMA channel shall us this register to store its state vector and other flag bits used during its active or idle period. This register is intended for debug purposes only.

Bit No.	Bit Name	Dir	Description
31 - 24	STATE_VEC[7:0]	r	State vector of the DMA channel. The current state of the main DMA control state machine.
23 - 21	unused	r	returns 0 when read
20	LOCK	r	The DMA state machine is executing a sequence of states which can not be interrupted by a higher priority channel.
19 - 16	LIST_OFFSET[4:0]	r/w	Current list offset. When written to in test mode (Test enable and channel n selected in the Test Register) the current list offset will increment.
15	STATE FLAG	r	Miscellaneous flag used by the state machine.
14 - 06	unused	r	returns 0 when read
05 - 00	CURRENT CONTEXT	r	Current channel context selected by the Priority Encoder and executing by the State Machine.

6.3.8 DMA Diagnostic Test Control @900

This register shall provide an interface for software to setup and perform diagnostic testing of the DMA control logic.

<table>
<tr><th>Bit No.</th><th>Bit Name</th><th>Dir</th><th>Description</th></tr>
<tr><td>31-24</td><td>MASTER BYTE COUNT</td><td>r</td><td>This is the computed number of bytes that the DMA will request during a PCI master cycle. The master byte count is equal to the lesser of the current HIGH WATER MARK, the current receive transfer count, and the DBXXFRLEN bits of the DMA channel control word. The selected channel is determined by the Channel Select bits of this register.</td></tr>
<tr><td>23-16</td><td>HIGH WATER MARK</td><td>r</td><td>This is the computed FIFO threshold which must be met before the DMA will request a PCI master cycle. In general, it is equal to HIGH WATER MARK which is equal to the greater of the cache line size +3 or the lower bound field of the DMA Global Register. In those cases where the HIGH WATER MARK value falls outside of the upper and lower bounds of 16 and 128 bytes. The lowest high water mark is 19 bytes, the highest is 131 bytes.</td></tr>
<tr><td>15</td><td>0</td><td>r</td><td>Return 0 when read.</td></tr>
<tr><td>14</td><td>ADDERTEST</td><td>r/w</td><td>Puts the DMA in a mode where the adder logic for the complete count bits of the Channel Status register, the list offset bits of the Current State register, and the remaining count bits of the Receive Packet Count register are in a test mode. The behavior of the registers are as follows:
Status register complete count bits: Any slave write to these bits while in the test mode will cause the current master byte count to be added to the contents of this register.
Current State list offset bits: Any slave write to these bits while in the test mode will cause these bits to increment.
Receive Packet Count register: Any slave write to these bits while in the test mode will cause the current master byte count to be subtracted from the contents of this register.
The value of master byte count is the lesser of the high water mark (cache line size +3), the receive transfer count register, and the DBXXFRLEN bits of the DMA channel control register.</td></tr>
<tr><td>13</td><td>DMA Test Mux Select Out</td><td>r</td><td>Read back of the signal selected by the above DMA test mux.</td></tr>
<tr><td>12-08</td><td>DMA Test Mux Sel [4:0]</td><td>r/w</td><td>Select DMA signal for to drive input mux of PCI Test Register.
<table>
<tr><th>Sel 4</th><th>Sel 3</th><th>Sel 2</th><th>Sel 1</th><th>Sel 0</th><th>DMA Signal</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>rcv_link_active</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>itf_link_active</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>atf_link_active</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>async_xmt_pkt_cnt_1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>rcv_pkt_cnt_gt_0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td><td>async_xmt_ok</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>0</td><td>iso_xmt_ok</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>iso_xmt_in_progress</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>rcv_fifo_empty</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>iso_xmt_wait</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>0</td><td>xmt_p1394rty</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>1</td><td>rcv_req</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>0</td><td>active_selq</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>1</td><td>idle_selq</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>0</td><td>rcv_link_active</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td></td><td></td><td></td><td>:</td><td></td><td></td><td>rcv_link_active</td></tr>
<tr><td></td><td></td><td></td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>rcv_link_active</td></tr>
<tr><td>07</td><td>unused</td><td>r</td><td colspan="7">Returns 0 when read.</td></tr>
<tr><td>06 - 01</td><td>CHANNEL SELECT[5:0]</td><td>r/w</td><td colspan="7">Select the DMA Channel for test. These bits select which channel context is selected when bit 00 DMATESTEN is set. The priority selection logic forces the selected context. One may then write to the current state bits of the current state register and force state machine execution starting at the loaded state.
CHANNEL SELECT[5:0] | DMA Channel Number
0 0 0 0 0 0 | 0
0 0 0 0 0 1 | 1
: | :
0 0 0 1 0 0 | 4
others | reserved</td></tr>
<tr><td>00</td><td>DMATESTEN</td><td>r/w</td><td colspan="7">Enable DMA diagnostic test mode. enable = 1 disable = 0. When enabled,
all DMA channel registers are readable and writable from the PCI bus.</td></tr>
</table>

6.3.9 Receive Packet Remaining Count Register @904

This register contains the current received packet count. The DMA loads this register with the receive packet count passed in the receive FIFO token words. This count is then decremented as the data is transferred to the PCI bus.

Bit No.	Bit Name	Dir	Description
31-13	unused	r	Return 0 when read.
12-00	REMAINING PACKET COUNT	r	The current remaining packet count.

6.3.10 Global Register @908

This register contains some flags and misc information the DMA uses during an asynchronous transmit operation.

Bit No.	Bit Name	Dir	Description
31- 30	unused	r	Return 0 when read.
29-24	LOWER BOUND	r/w	The burst transfer count requested by the DMA of the PCI master will be the greater of the value of these bits or the value of the cache line size register. The value of this register is expressed as a number of quadlets to transfer per burst request. These bits will be set to 0x4 upon reset.
23 - 03	unused	r	Return 0 when read.
02	FIFO FLUSH	r	Request the link to flush the async transmit FIFO.
01	PREVIOUS VALID	r	A PCL is currently pipelined and awaiting a status update.
00	RETRY	r	An ASYNC transmit retry is in progress.

6.4 FIFO Control and Status Register Definitions

6.4.1 FIFO Size @A00

This register shall be implemented to provide the application software with an interface for programming the size of each of the Logical FIFO's described in section 5.2.5, page 49 of this specification. Each FIFO shall be programmable in size from 0 to 255 quadlets. For any given combination, the sum of all 3 FIFO sizes shall be less than or equal to 256 quadlets. This register shall be initialized to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31 - 24	Not used	r	All zeros returned on a read
23 - 16	ITF_FIFOSZ[7:0]	r/w	isochronous transmit FIFO size 0x00 <= size <= 0xff
15 - 08	ATF_FIFOSZ[7:0]	r/w	Asynchronous transmit FIFO size 0x00 <= size <= 0xff
07 - 00	GRF_FIFOSZ[7:0]	r/w	General receive FIFO size. 0x00 <= size <= 0xff

6.4.2 PCI-Side FIFO Pointer Write-Read Port @A04

This register shall be implemented to provide the application software with a PCI slave access port for writing to or reading from the FIFO pointers which are used in accessing the PCI-side of the FIFO..

Bit No.	Bit Name	Dir	Description
31 - 27	Not used	r	All zeros returned on a read
26	ITF_WAB_L	r/w	PCI-side ITF write pointer Wrap-around bit
25	ATF_WAB_L	r/w	PCI-side ATF write pointer Wrap-around bit
24	GRF_WAB_L	r/w	PCI-side GRF read pointer Wrap-around bit
23 - 16	PCI_ITF_WPTR[7:0]	r/w	PCI-side isochronous transmit FIFO write pointer value returned on read = grf pointer contents
15 - 08	PCI_ATF_WPTR[7:0]	r/w	PCI-side asynchronous transmit FIFO write pointer value returned on read = grf pointer contents + (ITF_size)
07 - 00	PCI_GRF_RPTR[7:0]	r/w	PCI-side general revive FIFO read pointer. value returned on read = grf pointer contents + (ATF_ size + ITF_size)

6.4.3 Link-Side FIFO Pointer Write-Read port @A08

This register shall be implemented to provide the application software with a PCI slave access port for writing to or reading from the FIFO pointers which are used in accessing the Link-side of the FIFO.

Bit No.	Bit Name	Dir	Description
31 - 27	Not used	r	All zeros returned on a read
26	ITF_WAB_L	r/w	Link-side ITF read pointer Wrap-around bit
25	ATF_WAB_L	r/w	Link-side ATF read pointer Wrap-around bit
24	GRF_WAB_L	r/w	Link-side GRF write pointer Wrap-around bit
23 - 16	LINK_ITF_RPTR[7:0]	r/w	LINK-side isochronous transmit FIFO read pointer value returned on read = ITF pointer contents
15 - 08	LINK_ATF_RPTR[7:0]	r/w	LINK-side asynchronous transmit FIFO read pointer value returned on read = ATF pointer contents + (ITF_size)
07 - 00	LINK_GRF_WPTR[7:0]	r/w	LINK-side general receive FIFO write pointer value returned on read = GRF pointer contents + (ATF_ size + ITF_size)

6.4.4 FIFO Control Token Status Read-Port @A0C

This port shall be implemented to provide an interface for application software to obtain the value of bit 32(MSB) of the 33 bit data data value from the last fifo that was popped.

Bit No.	Bit Name	Function
31-02	Not Used	All zero's returned on a read
01	GRF_FCT32	bit 32 data value of last pop operation from the GRF
00	TF_FCT32	bit 32 data value of last pop operation from the ITF or ATF

6.4.5 FIFO Control and test Register @A10

This register shall be implemented to provide the application software with an interface for test control and flushing of a selected FIFO. This register shall be initialized to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31 - 10	not used	r	return 0 for these bits on a read
11 - 08	TEST_MUX[3:0]	r/w	Test mux. Selects 1 of 16 test points within the FIFO logic for observation at external TEST_OUT pin TEST_MUX [3:0] — Test Point Selected hex0 — GRF output register data bit32 hex1 — ATF output register data bit32 hex2 — ITF output register data bit 32 hex3-F — To be Assigned Later. logic 0 will be outputted
07 - 05	not used	r	0 returned on a read for these bits
04	GRF_FLUSH	r/w	GRF flush. When set to a 1, this bit will flush the contents of the GRF by setting the GRF read and write pointers to 0. This bit shall be self-clearing
03	ITF_FLUSH	r/w	ITF flush. When set to a 1, this bit will flush the contents of the ITF by setting the ITF read and write pointers to 0. This bit shall be self-clearing
02	ATF_FLUSH	r/w	ATF flush. When set to a 1, this bit will flush the contents of the ATF by setting the ATF read and write pointers to 0. This bit shall be self-clearing
01	FORCE_BIG_ENDIAN	r/w	When set to a Logic 1, Slave data written to the ATF or ITF will be stored in big endian byte order (bytes are not swapped). When set to a 0, data will be stored in little endian order (bytes are swapped).
00	FCT33_WR	r/w	The 32 bit PCI slave write data that will be pushed onto a selected FIFO will be pushed to fifo bit positions (31 - 00). The current value of FCT33_WR will be pushed into bit position 32. When FCT33_WR = 1, the data pushed to bits (31- 00) will be interrepted as a FIFO control token. FCT33_WR = 0 indicates that data pushed to bits (31 - 00) are normal data.

6.4.6 Asynchronous and Isochronous Transmit FIFO Threshold Control @A14

This register shall be implemented to provide the application software with an interface for setting the ATF or ITF data ready threshold levels. When the number of data quadlets written into the ATF or ITF is greater than or equal to threshold setting, a ready signal is issued to the 1394 link transmitter which indicates that the ATF or ITF has enough data to begin a packet transmission. This register shall be initialized to 0x0000ffff on power-up reset.

Bit No.	Bit Name	Dir	Description
31 - 16	Reserved	r	0's returned in these bit positions on a read
15 - 08	ATF_TRSHLD[7:0]	r/w	ATF Transmit ready threshold in bytes. Valid range= hex00 to hexff
07 - 00	ITF_TRSHLD[7:0]	r/w	ITF Transmit ready threshold in bytes. Valid range= hex00 to hexff

6.4.7 General Receive FIFO Push-Pop Ports @A20 A24

This port shall be implemented to provide an interface for application software to write and-or read 32 bit data qaudlets to-from from the 33 bit wide General Receive FIFO, via a PCI slave access. A write (PUSH port 0) to address offset 0xA20 shall cause the 32 bit data quadlet to be written to the 33 bit wide FIFO memory location with MSB bit 32 set to a 0 and the 32 bit data quadlet written to bits 31 to 00. A write (PUSH port 1) to address offset 0xA24 shall cause the 32 bit data quadlet to be written to the 33 bit wide FIFO memory location with MSB bit 32 set to a 1 and the 32 bit data quadlet written to bits 31 to 00. A read (POP) from either address offset will return to the software the data quadlet stored in bits 31-00 of FIFO memory location being accessed. The read shall cause MSB bit 32 of the FIFO memory location to be stored in the FIFO control token status register at offset 0xA0C . Software can then read the control token status register to determine the value that was read from bit 32 of of the FIFO memory location. Bit 32 can also be switched to the test mux output of the chip by programming the FIFO and PCI macro test muxes to select this bit for observation. When the read (POP) is performed, the value of bit 32 will appear at the test mux output of the CHIP during the active portion of the slave read cycle.

6.4.8 Asynchronous Transmit FIFO Push-Pop Ports 0 and 1 @A30 A34

This port shall be implemented to provide an interface for application software to write and-or read 32 bit data qaudlets to-from from the 33 bit wide Asynchronous Transmit FIFO, via a PCI slave access. A write (PUSH port 0) to address offset 0xA30 shall cause the 32 bit data quadlet to be written to the 33 bit wide FIFO memory location with MSB bit 32 set to a 0 and the 32 bit data quadlet written to bits 31 to 00. A write (PUSH port 1) to address offset 0xA34 shall cause the 32 bit data quadlet to be written to the 33 bit wide FIFO memory location with MSB bit 32 set to a 1 and the 32 bit data quadlet written to bits 31 to 00. A read (POP) from either address offset will return to the software the data quadlet stored in bits 31-00 of FIFO memory location being accessed. The read shall cause MSB bit 32 of the FIFO memory location be stored in the FIFO control token status register at offset 0xA0C . Software can then read the control token status register to determine the value that was read from bit 32 of of the FIFO memory location. Bit 32 can also be switched to the test mux output of the chip by programming the FIFO and PCI macro test muxes to select this bit for observation. When the read (POP) is performed, the value of bit 32 will appear at the test mux output of the CHIP during the active portion of the slave read cycle.

6.4.9 Isochronous Transmit FIFO Push-Pop Ports 0 and 1 @A40 A44

This port shall be implemented to provide an interface for application software to write and-or read 32 bit data qaudlets to-from from the 33 bit wide Isochronous Transmit FIFO, via a PCI slave access. A write (PUSH port 0) to address offset 0xA30 shall cause the 32 bit data quadlet to be written to the 33 bit wide FIFO memory location with MSB bit 32 set to a 0 and the 32 bit data quadlet written to bits 31 to 00. A write (PUSH port 1) to address offset 0xA34 shall cause the 32 bit data quadlet to be written to the 33 bit wide FIFO memory location with MSB bit 32 set to a 1 and the 32 bit data quadlet written to bits 31 to 00. A read (POP) from either address offset will return to the software the data quadlet stored in bits 31-00 of FIFO memory location being accessed. The read shall cause MSB bit 32 of the FIFO memory location to be stored in the FIFO control token status register at offset 0xA0C . Software can then read the control token status register to determine the value that was read from bit 32 of of the FIFO memory location. bit 32 can also be switched to the test mux output of the chip by programming the FIFO and PCI macro test muxes to select this bit for observation. When the read (POP) is performed, the value of bit 32 will appear at the test mux output of the CHIP during the active portion of the slave read cycle.

6.5 1394 Link Layer Control and Status Register Definitions

6.5.1 DMA Channel 0 - 4 Word 0 Receive Packet Compare Value Register @B00 B10 B20 B30 B40

This register shall be implemented to provide the interface for application software to program the compare-to-value which shall used by the channel address comparator logic to match against the first word in the received packet. Bit position 31 is the MSB.

Bit No.	Bit Name	Dir	Description
31 - 16	CMP0_FIELD1[15:0]	r/w	Specifies a 16 bit value to match against the destination ID field of an incoming ASYNC or ISO packet
15 - 08	CMP0_FIELD2[7:0]	r/w	Specifies an 8 bit value to match against the transaction label and retry fields on an incoming ASYNC packet or the CHANNEL number field of an incoming ISO packet.
07 - 04	CMP0_FIELD3[3:0]	r/w	Specifies an 4 bit value to match against the tcode field on an incoming ASYNC or ISO packet.
03 - 00	CMP0_FIELD4[3:0]	r/w	Specifies a 4 bit value to match against the PRIORITY field of an incoming ASYNC packet or the SYSTEM field of an incoming ISO packet

6.5.2 DMA Channel 0 - 4 Word 0 Receive Packet Compare Enable Register @B04 B14 B24 B34 B44

This register shall be implemented to provide the interface for software to program the address comparator field select mask. This value shall be use to specify the bit fields that will be checked by the comparator logic when matching the value of the word 0 comparator register to the first word of the incoming packet. Bit position 31 is the MSB.

Bit No.	Bit Name	Dir	Description
31 - 16	CMP0_FIELD1 _ MASK[15:0]	r/w	Specifies a 16 bit enable value to select the CMP0_FIELD1[15:0] bits for matching against the destination ID field of an incoming ASYNC packet. A 0xFFFF mask value enables the match. 0x0000 disables the match.
15 - 08	CMP0_FIELD2 _MASK[7:0]	r/w	Specifies an 8 bit mask value to select the CMP0_FIELD2[7:0] bits for matching against the transaction label and retry fields of an incoming ASYNC packet, or the channel no. field of an ISO packet A 0xFF mask enables the match. 0x00 disables the match.
07 - 04	CMP0_FIELD3 _MASK[3:0]	r/w	Specifies a 4 bit code for programming the tcode compare function mode which shall select the GRF for receiving the incoming packet. 0000 - Disable tcode compariaion always equal -> GRF 0001 - ISO tcode matches CMP0_FIELD3[3:0] -> GRF 0010 - ISO tcode does not match CMP0_FIELD3[3:0] -> GRF 0011 - ASYNC tcode matches CMP0_FIELD3[3:0] -> GRF 0100 - ASYNC tcode does not match CMP0_FIELD3[3:0] -> GRF 0101 - New ISO tcode match CMP0_FIELD3[3:0] + “normal iso encode” -> GRF 0110 - New ASYNC tcode match CMP0_FIELD3[3:0] + “normal async encode” -> GRF 0111 - New ISO tcode does not match CMP0_FIELD3[3:0] + “normal iso encode” -> GRF 1000 - New ASYNC tcode does not match CMP0_FIELD3[3:0] + “normal async encode” -> GRF 1001 - 1394 Iso tcode encoding (“normal”) -> GRF 1010 - 1394 Async tcode encoding (“normal”) -> GRF 1011 - 1111 Reserved - always equal

03 - 00	CMP0_FIELD4 _MASK[3:0]	r/w	Specifies a 4 bit mask value to select the CMP0_FIELD4[3:0] for matching against the PRIORITY field of an incoming ASYNC packet or the SYSTEM field of an incoming ISO packet. A 0xF mask enables the match. 0x0 disables the match.

6.5.3 DMA Channel 0 - 4 Word 1 Receive Packet Compare Value Register @B08 B18 B28 B38 B48

This register shall be implemented to provide the interface for application software to program a compare-to-value which shall used by the channel address comparator logic to match against the second word in an incoming packet. This register shall be initialized to 0x00000000 on power-up reset. Bit position 31 is the MSB.

Bit No.	Bit Name	Dir	Description
31 - 16	CMP1_FIELD1[15:0]	r/w	Specifies a 16 bit value to match against the source ID field of an incoming ASYNC packet
15 - 00	Reserved	r	Return 0's on read

6.5.4 DMA Channel 0 - 4 Word 1 Receive Packet Compare Enable Register @ B0C B1C B2C B3C B4C

This register shall be implemented to provide the interface for software to program the address comparator field select mask. This value shall be use to specify the bit fields that will be checked by the comparator logic when matching the value of the word 1 comparator register to the second word of the incoming packet. EN_CH_COMPARE and WRITE_REQ_ACK_SEL bits shall be initialized to 0 on power-up reset. Bit position 31 is the MSB

Bit No.	Bit Name	Dir	Description
31 - 16	CMP1_FIELD1_MASK[15:0]	r/w	Specifies a 16 bit mask value to select the CMP1_FIELD1[15:0] bits for matching against the source ID field of an incoming ASYNC packet.
15 - 11	DEST_ID_SEL[4:0]	r/w	Specifies the operating mode of the destination ID comparator logic. packet_wdo[31:0] is the first quadlet of the incoming packet. xxxx1 match packet_wd0[31:22] to Bus_Number register and packet_wd0[21:16] to Node_Number register xxx1x match packet_wd0[31:22] to 3FF and packet_wd0[21:16] to Node_Number register xx1xx match packet_wd0[31:22] to Bus_Number register and packet_wd0[21:16] to 3F x1xxx match packet_wd0[31:22] to 3FF packet_wd0[21:16] to 3F 1xxxx match packet_wd0[31:22] to not equal to Bus_Number register and packet_wd0[31:22] to not equal to hex3FF
10	RCV_SELF_ID_EN	r/w	Enable reception of self-ID packets. Enable = 1 Disable = 0
09	EN_DIRECT_ADR	r/w	If bits 15-0 of word 1 are all zeros, the destination offset specified in word 2 will be used by the DMA controller as the starting address in PCI memory space for the data transfer operation specified by the incoming async packet. enable = 1. disable = 0
08	EN_CH_COMPARE	r/w	Channel comparator master enable. Enable = 1 channel comparator is enabled for normal operation. Disable = 0 channel comparator always returns a no-match indication on incoming packet.
07	WRITE_REQ_ACK_SEL	r/w	Write Request Acknowledge select. When set to 1 an incoming non-

			broadcast write request packet will be ack'ed with an ack complete (hex0001). When set to 0 ack pending will be used (hex0010).
06 - 00	Reserved	r/w	Return 0's on a read

6.5.5 Bus Number and Node Number @F00

This register shall be implemented to provide the interface for application software to program the 1394 bus and node identification numbers that are assigned to the PCI-LYNX by the 1394 bus management layer. This register shall initialize to 0x00000000 on power-up reset. Bit position 31 is the MSB.

Bit No.	Bit Name	Dir	Description
31 - 22	BUS_ID[9:0]	r/w	1394 bus identification number
21 - 16	NODE_ID[5:0]	r/w	1394 Node identification number
15 - 00	Reserved	r	return 0's for this bits on a read

6.5.6 1394 Link layer Control @F04

This register shall be implemented to provide the interface for application software to program the operation of the 1394 link layer control logic for controlling the transmission and reception of 1394 data packets. This register shall initialize to 0x00000000 on power-up reset. Bit position 31 is the MSB.

Bit No.	Bit Name	Dir	Description
31 - 30	reserved	r	return 0's on a read
29	BUSY_CNTRL	r/w	Controls what busy status the LLC will return on incoming packets that cannot be received. **0 =** Use single phase busy protocol to busy incoming packets addressed to this node only when the GRF is unavailable. **1 =** Use single phase busy protocol to unconditionally busy all incoming packets addressed to this node until software sets this bit to 0.
28 - 27	Reserved	r	Return 0's on a read
26	TX_ISO_EN	r/w	Enable transmitter to send 1394 ISO packets. enable = 1 disable = 0
25	RX_ISO_EN	r/w	Enable receiver to receive 1394 ISO packets. enable = 1 disable = 0
24	TX_ASYNC_EN	r/w	Enable transmitter to send 1394 ASYNC packets. enable = 1 disable = 0
23	RX_ASYNC_EN	r/w	Enable receiver to receive 1394 ASYNC packets. enable = 1 disable = 0
22	Reserved	r	Return 0's on a read
21	RSTTX	r/w	Reset 1394 transmitter. Reset = 1 causes synchronous reset of transmitter logic. This bit shall be self-clearing.
20	RSTRX	r/w	Reset 1394 receiver. Reset = 1 causes synchronous reset of receiver logic. This bit shall be self-clearing.
19 - 12	Reserved	r	Return 0's on a read
11	CYCMASTER	r/w	Enable PCI-LYNX to be the cycle master. When set to 1 and the PCI-LYNX is attached to the ROOT PHY, the LLC 1394 transmit logic shall send a cycle start packet each time the cycle count field of the cycle timer register enable increments
10	CYCSOURCE	r/w	Enable cycle source. When set to 1, the cycle count field of the cycle timer register will increment and the cycle offset field will reset for each rising edge of transition applied to the CYCLIN pin of the PCI-LYNX ASIC. When set to 0. the cycle_count field will increment when the cycle_offset field rolls over.
09	CYCTIMEREN	r/w	Enable cycle timer to increment. enable = 1 disable = 0
08	Not used	r/w	0 returned on read
07	RCV_COMP_VALID	r/w	RCV_COMP_VALID = 1. Bus number-node number register and rcv comparator registers have been programmed with valid data.
06-00	not used	r	Return 0's on a read

6.5.7 1394 Cycle Timer @F08

This register shall be implemented to provide the interface for application software to program 1394 cycle timer counters with an initial value or read the current state of the counters. This register shall initialize to 0x00000000 on power-up reset. Bit position 31 is the MSB.

Bit No.	Bit Name	Dir	Description
31 - 12	CYCLE_NUMBER[19:0]	r/w	ISO Cycle number . Increments every 125usec
11 - 00	CYCLE_OFFSET[11:0]	r/w	24.576 MHz cycle timer counter - cycle offset rollsover every 125usec

6.5.8 1394 Physical layer Access F0C

This register shall be implemented to provide the interface for application software to access the control and status registers located in the Physical layer chip. This register shall be initialized to 0x000000000 on power-up reset. Bit position 31 is the MSB.

Bit No.	Bit Name	Dir	Description
31	RDPHY	r/w	Read PHY register request. When set to 1 the LLC logic shall send a read request to the PHY to return the value of the PHY register whose address is specified by bit field 07-04 of this register definition. This bit shall be self-clearing.
30	WRPHY	r/w	Write PHY register request. When set to a 1, the LLC logic shall send a write request to the PHY layer to write the 8 bit values specified in bit field 15-08, to the PHY address specified in bit field 07-04. This bit shall be self-clearing.
29 - 28	not used	r	return 0 for these bits on a read
27 - 24	PHY_REG_ADR[3:0]	r/w	Address of the PHY register to be written to or read from. valid address from 0x0 to 0x8
23 - 16	PHY_REG_DAT[7:0]	r/w	Data to be written to the PHY register specified in bit field 07-04
15 - 12	not used	r	Return 0 for these bits on a read
11 - 08	PHY_REGRD_ADR[3:0]	r/w	Address of PHY register which was read. This address is returned by the PHY in the status message in sends in response to a PHY register read request.
07 - 00	PHY_REGRD_DAT[7:0]	r/w	Data read from a selected PHY register. This data is returned by the PHY in the status message in sends in response to a PHY register read request.

6.5.9 1394 Diagnostic Test Control @F10

This register shall be implemented to provide the interface for application software to perform diagnostic testing of the 1394 LLC functionality. This register shall initialize to 0x00000000 on power-up reset.

Bit No.	Bit Name	Dir	Description
31 - 15	Reserved	r	Return 0's on read
14	CH_MATCH	r	This test point shall be the channel match indication generated by the link core address comparator logic
13 - 08	DMA_CH_NO[5:0]	r	This test point shall be the 6 bit DMA channel number generated by the link core address comparator logic
07	GRF_UFLOW_TST_STB	r/w	GRF underflow counter test increment. When a 1 is written to the Bit, a one clock wide pulse shall be generated to the GRF underflow counter. This pulse shall cause the counter to increment by 1. This bit shall be self-clearing
06	ATF_UFLOW_TST_STB	r/w	ATF underflow counter test increment. When a 1 is written to the Bit, a one clock wide pulse shall be generated to the ATF underflow counter. This pulse shall cause the counter to increment by 1. This bit shall be self-clearing
05	ITF_UFLOW_TST_STB	r/w	ITF underflow counter test increment. When a 1 is written to the Bit, a one clock wide pulse shall be generated to the ITF underflow counter. This pulse shall cause the counter to increment by 1. This bit shall be self-clearing
04 - 01	TESTMUXSEL[3:0]	r/w	Select internal test point for observation at the external TEST_OUT pin
00	DIAG1394EN	r/w	Enable 1394 diagnostic test mode. When set to 1, the 1394 diagnostic test mode is enabled. Set to 0 to enable normal operating mode. The setting of this bit to 1 shall enable the specific diagnostic test modes defined in this register definition

6.5.10 1394 Link Layer Interrupt Status Register @F14

This register shall be implemented to provide the interface for application software to determine the interrupt that is caused by a 1394 link event. This register shall be set to 0x00000000 on power up reset. An interrupt bit shall be asserted when it is set to a logic 1. The interrupt Status bits defined in the following tabled shall be cleared by writing a "1" to a selected bit position. Bit position 31 is the MSB

Bit No.	Bit Name	Dir	Description
31	LINK_INT	r/w	Link logic interrupt. This signal is the "OR" of all link interrupt sources.
30	PHY_TIME_OUT	r/w	The Phy has stayed in a particular state for to long.
29	PHY_REG_RCVD	r/w	The contents of a Phy register has been received from the Phy.
28	PHY_BUSRESET	r/w	The Phy has entered the bus reset state.
27	Reserved	r	Return 0's on a read
26	TX_RDY	r/w	The transmitter is ready to transmit a packet.
25	RX_DATA_RDY	r/w	The receiver has received a packet.
24 - 21	Reserved	r	Return 0's on a read
20	IT_STUCK	r/w	Transmitter stuck on ISO transfer from ITF.
19	AT_STUCK	r/w	Transmitter stuck on ASYNC transfer from ATF.
18	Reserved	r	Return 0's on a read
17	SNTRJ	r/w	The receiver was force to send a busy acknowledge to a packet addressed to this node because the GRF overflowed.
16	HDR_ERR	r/w	The receiver detected a header CRC error on an icoming packet that may have been addressed to this node.

15	TC_ERR	r/w	the Transmitter detected an invalid transaction code in the packet that it was attempting to transmit.
14 - 12	Reserved	r	Return 0's on a read
11	CYC_SEC	r/w	Cycle timer in seconds has incremented.
10	CYC_STRT	r/w	Cycle start packet was sent or received.
09	CYC_DONE	r/w	A sub-action gap has been detected on the bus after the transmission or reception of a cycle start packet. This indicates that the ISO cycle is over.
08	CYC_PEND	r/w	Cycle pending is asserted when cycle timer offset is set to zero (rolled over or reset) and statys asserted until the ISO cycle has ended.
07	CYC_LOST	r/w	Cycle timer has rolled over twicw without receiving a cycle start packet.
06	CYC_ARB_FAILED	r/w	The arbitration to send the cycle start packet has failed.
05	GRF_OVER_FLOW	r/w	GRF over_flow detected during packet reception
04	ITF_UNDER_FLOW	r/w	ITF under flow detected during ISO packet transmission
03	ATF_UNDER_FLOW	r/w	ATF underflow detected during ASYNC packet transmission
02 - 01	Reserved	r/w	Return 0's on a read
00	IARB_FAILED	r/w	Arbitration to send an ISO packet has failed.

6.5.11 1394 Link Layer Interrupt Enable Register @F18

This register shall be implemented to provide the interface for application software to enable the interrupt specified in the 1394 link layer interrupt status register. This register shall be set to 0x00000000 on power up reset. Bit position 31 is the MSB. Setting an enable bit to a logic 1 enables the interrupt. Setting the enable bit to logic 0 disables the interrupt.

Bit No.	Bit Name	Dir	Description
31	Reserved	r	Return 0 on a read
30	PHY_TIME_OUT_EN	r/w	Enable Phy time out interrupt.
29	PHY_REG_RCVD_EN	r/w	Enable Phy register data received interrupt.
28	PHY_BUSRESET_EN	r/w	Enable Phy bus reset interrupt.
27	Reserved	r	Return 0's on a read
26	TX_RDY_EN	r/w	Enable Transmitter sent packet interrupt.
25	RX_DATA_RDY_EN	r/w	Enable Receiver received packet interrupt.
24 - 21	Reserved	r	Return 0's on a read
20	IT_STUCK_EN	r/w	Enable Transmitter stuck on ISO interrupt.
19	AT_STUCK_EN	r/w	Enable Transmitter stuck on ISO interrupt.
18	Reserved	r	Return 0's on a read
17	SNTRJ_EN	r/w	Enable receiver sent busy ack interrupt.
16	HDR_ERR_EN	r/w	Enable receiver header error interrupt.
15	TC_ERR_EN	r/w	Enable transmitter invalid Tcode interrupt.
14 - 12	Reserved	r	Return 0's on a read
11	CYC_SEC_EN	r/w	Enable cycle timer seconds interrupt.
10	CYC_STRT_EN	r/w	Enbale cycle start interrupt.
09	CYC_DONE_EN	r/w	Enable cycle done interrupt.
08	CYC_PEND_EN	r/w	Enable cycle pending interrupt.
07	CYC_LOST_EN	r/w	Enable cycle lost interrupt.
06	CYC_ARB_FAILED_EN	r/w	Enable cycle start arbitration failed interrupt.
05	GRF_OVER_FLOW_EN	r/w	Enable GRF over flow interrupt
04	ITF_UNDER_FLOW_EN	r/w	Enable ITF under flow interrupt
03	ATF_UNDER_FLOW_EN	r/w	Enable ATF under flow interrupt
02 - 01	Reserved	r	Return 0's on a read
00	IARB_FAILED_EN	r/w	Enable ISO arb failed interrupt.

6.5.12 1394 Busy Retry Control Register @F1C

This register shall be implemented to provide the interface for application software to set the number of times that the 1394 transmitter is to retry an asynchronous packet that was ack'ed with a busy or error condition. This register also provides the the means to program the time interval to delay between succesive retrys. Bit position 31 is the MSB. This register shall be cleared to all zeros on power-up reset. Bit position 31 is the MSB.

Bit No.	Bit Name	Dir	Description
31-16	Reserved	r	Returns all zero's on read.
15-08	BUSY_RETRY_DLY[7:0]	r/w	A number between 0 and 255 that shall specify the time that the 1394 transmitter must delay between successive retries. This time shall be equal to BUSY_RETRY_DLY[7:0] times ISO_INTERVAL(125usec)
07-00	BUSY_RETRY_CNT[7:0]	r/w	A number between 0 and 255 that shall specify the maximum number of times to re-transmit a packet, when the destination node continues to return busy acknowledge status. The 1394 transmitter shall notify the active DMA channel when the maximum number of transmit retries have been attempted without a successful transmission occurring.

6.5.13 Link Layer Controller State Machine Vector Monitor Port @F20

This register shall be implemented to provide application software with an I/O port to read the value of the state vector for each state machine in the Link Layer Control logic. This register is read only.

Bit No.	Bit Name	Dir	Description
31 - 28	TRANSMIT_IFC_STATE[3:0]	r	TransmitIfc state machine vector
27 - 25	RXSTATUS_IFC_STATE[2:0]	r	RxdStatIfc state machine vector
24 - 22	RXDDATA_IFC_STATE[2:0]	r	RxdDataIfc state machine vector
21 - 19	RCV_ACK_STATE[2:0]	r	RcvAck state machine vector
18 - 15	LREQ_STATE[3:0]	r	Request state machine vector
14 - 09	RECEIVE_STATE[5:0]	r	Receive state machine vector
08 - 03	TRANSMIT_STATE[5:0]	r	Transmit state machine vector
02 - 00	CM_STATE[2:0]	r	CycleMonitor state machine vector

6.5.14 Link Layer FIFO Under Flow - Over Flow Counters @F24

These counters shall be implemented to provide application software with an I/O port to monitor the number of ATF and ITF underflows that have occurred during packet transmissions and the number of over flows that have occurred during packet reception. These counters are cleared to logic 0's on power-up reset. Bit 31 is MSB

Bit No.	Bit Name	Dir	Description
31 - 24	Not used	r	0's returned on read
23 - 16	ITF_UNDER_FLOW[7:0]	r/w	Increments by 1 on each ITF under flow detected by 1394 transmitter or when a 1 is written to the test increment bit in the diagnostic test register. Counting Stops and holds when counter == hexFF. Subsequent undeflows will not increment the counter. The counter is enabled when software loads it with any value that is less than hexFF. Counting will proceed from that value until hexFF is reached.
15 - 08	ATF_UNDER_FLOW[7:0]	r/w	Increments by 1 on each ATF under flow detected by 1394 transmitter. Counting Stops and holds when counter == hexFF. Subsequent undeflows will not increment the counter. The counter is enabled when software loads it with any value that is less than hexFF. Counting will proceed from that value until hexFF is reached.
07 - 00	GRF_OVER_FLOW[7:0]	r/w	Increments by 1 on each GRF over flow detected by 1394 transmitter. Counting Stops and holds when counter == hexFF. Subsequent undeflows will not increment the counter. The counter is enabled when software loads it with any value that is less than hexFF. Counting will proceed from that value until hexFF is reached.

7. APPENDIX A - Design Methodology and Conventions

7.1 PURPOSE

The purpose of this appendix is to suggest uniform design methodologies and conventions for the PCI-LYNX ASIC development. This should facilitate the development team working together in an efficient and cohesive manner.
The general intent of the conventions is to improve the readability and understandability of the source code. To this end, meaningful names and commented code should be emphasized.
Major logic blocks of source code imported into this design from other sources may not conform to this document. Such source files may be changed to conform if it is convenient and useful to change.

7.2 Revision Control

Source file revision control is maintained by a manual system of appending a final ".n" to the file name. The "n" is a sequential number representing the revision level of the file. Thus, revision "0" of a file will be named "filename.v.0", the next revision, "filename.v.1", "filename.v.2", "filename.v.3", and so forth. It is up to the designer to decide when changes dictate a new revision, and when out-dated revisions are deleted from the database.
With this scheme, multiple designers can continue to work with a known snapshot of each other's work while the modules all continue to evolve.

7.3 File Names and Hierarchy

There should be one Verilog module in each source file.
Verilog source file names should be the same as the Verilog module name with a ".v" suffix appended.
File names should use lower case letters only, and should use underscores ("_") to provide separation of phrases.
File names should be as descriptive as possible of the function of the module.
There is no maximum length for file names; however, over about 16 characters starts becoming awkward.
The current hierarchy is intended to be a basic structure to work from:

```
pci_link              /design         /pci            /verilog_src
                                                      /scripts
                                                      / .......
                                      /dma
                                      /fifo
                                      /link
                                      /top
                      /good_database
                      /spec
                      /schedule
```

7.4 Signal naming conventions

Signal and variable names should use lower case letters only, and should use underscores ("_") to provide separation of phrases.
Each signal name should be as descriptive as possible of its function.
Try to avoid signal names that are very similar or easily confused.
There is no maximum length for signal names; however, over about 10 characters starts becoming awkward and over about 16 becomes painful.
Low active signals should end with a "z" suffix. examples: resetz, loadz, data_busz[3:0].
Constants and parameters should use all upper case name, and should use underscores to provide separation of phrases.
Where it improves readability and understandability, (primarily to signify synchronization), a trailing "q" may be added to the signal name. example: rcv_dataqz, inputq
For most of the design, the global asynchronous reset "resetz" should be used. This signal is asynchronous on assertion, and synchronous with pck_clk on de-assertion. Each module should synchronize to the local clock domain, as required.

7.5 Coding Guidelines

7.5.1 Code Comments

7.5.1.1 File headers/body

```
//-----------------------------------------------------------//
// Description:         PCI Output Path Prefetch Registers
//
// Filename:    out_path_regs.v
//
// Author:      Brian Karguth
//              Richard  Baker
//
// Revision:    12/5/94 first version
//              12/12/94 rtb added comments
//-----------------------------------------------------------//
// Synthesis notes/critical pathes:
//-----------------------------------------------------------//
// Functional Description:
//
//-----------------------------------------------------------//
```

7.5.1.2 Signal definition:

```
//----- CONNECTING DEVICE SIGNALS (from connecting device -----//
input   load_mstr_cmd;          // load a cmd to the PCI MASTER
input [31:0] master_adr;        // PCI master address
input [5:0] mstr_xfr_length;    // Transfer byte count
input   mstr_req,               // Transfer request from DMA
        mstr_write,             // Write = 1 Read = 0 (alias wr_not_rd)
        big_endian;             // from DMA cntl word - do byte swap
input [31:0] master_data;       // PCI master write data

output mstr_read_en;    // reading new burst data to connecting device
//-----------------------------------------------------------//
```

7.5.1.3 Document logic:

```
// load output data pipeline:
//       swapped_mstr_data ==>next1_master_data==>curr_mast
```

7.5.1.4 Register definition:

```
//-------------------------------------------------------------//
// Address 0x44 -- Serial EEPROM Control Register [31:0]
//
//   31                          24 | 23                          16
//  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
//  | 0   0   0   0   0   0   0   0 | 0   0   0   0   0   0   0   0 |
//  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
//
//   15          12               8 | 7                           0
//  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
//  | 0   0   0   0   0   0   0   0 |NOT|EEP|EEP|EEP| 0   0 |10U| 5U|
//  |                               |PRS|CLK|ENA|DAT|       |TIM|TIM|
//  +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+
//
//-------------------------------------------------------------//
```

7.5.2 Coding style -- use/placement of begin/end

Generally, begin/end pairs should only be used where required.

7.5.3 Coding style - registers

Redundant code (i.e. else eep_cntl_reg[7:0] = eep_cntl_reg[7:0]; is not required and makes the source code harder to read.

```
//-------------------------------------------------------------//
// eep register
//-------------------------------------------------------------//
always @(posedge pb_clk or negedge resetz)
  begin
    if (~resetz)
      eep_cntl_reg <= #`DELAY 32'b0;
    else
      begin
        if (ltchd_addr[5:2] == 4'b0001 && cmd_type_wrz
                            && ~intreg_beginz && ~intreg_endz)
          begin
            if (~ltchd_bez[0]) // byte enable[0]
              eep_cntl_reg[7:0] <= #`DELAY p2slv_data[7:0];
          end
      end
  end
//-------------------------------------------------------------//
```

7.5.4 Coding style - state machines

State machine combinatorial next state logic should be in a separate process from the clocked process that loads the next_state to the current_state registers.

```
//-------------------------------------------------------------//
// state definitions
//-------------------------------------------------------------//
reg [3:0] current_state, next_state;
parameter        RESET   = 0;
parameter        CHECK   = 1;
parameter        OFFBUS  = 2;
parameter        START1B = 3;

//-------------------------------------------------------------//
// assignment of state register
//-------------------------------------------------------------//
always @(posedge pb_clk or negedge resetz)
  begin
    if (~resetz)
      current_state <= #`DELAY   RESET;
    else
      current_state <= #`DELAY next_state;
  end

//-------------------------------------------------------------//
// next state logic
//-------------------------------------------------------------//
always @(input1 or input2 or input3)
  begin
    case (current_state)

//==================//
// R E S E T
//==================//
      RESET:
        if (input1)
          next_state <= #`DELAY   CHECK;
//==================//
// C H E C K
//==================//
      RESET:
        next_state <= #`DELAY   OFFBUS;
// . . . . . .

    endcase

//-------------------------------------------------------------//
```

```
//-------------------------------------------------------------//
// synchronous outputs
//-------------------------------------------------------------//
always @(posedge pb_clk or negedge resetz)
  begin
    if (~resetz)
      outputs <= #`DELAY reset;
    else
      begin
        case (next_state)
           RESET:
             out1<=#`DELAY 0;out2<=#`DELAY 0;out3<=#`DELAY 0;out4<=#`DELAY 0;
           STATE1:
             out1<=#`DELAY 1;out2<=#`DELAY 0;out3<=#`DELAY 0;out4<=#`DELAY 0;
// . . . . . .

       endcase
// . . . . . .

// alternate to case statement
        if(next_state ==  RESET || next_state ==  EXIT)
          out1 <= #`DELAY 0;
        else
          out1 <= #`DELAY 1;
// . . . . . .

     end
  end

//-------------------------------------------------------------//
// asynchronous outputs
//-------------------------------------------------------------//
always @(in1 or in2 or in3 or in4)
  begin
    case (current_state)
         RESET:
          out1<=#`DELAY 0;out2<=#`DELAY 0;out3<=#`DELAY 0;out4<=#`DELAY 0;
         STATE1:
          out1<=#`DELAY 1;out2<=#`DELAY 0;out3<=#`DELAY 0;out4<=#`DELAY 0;
// . . . . . .

     endcase
  end
//-------------------------------------------------------------//
```

7.5.5 Clocks and Resets

As much as possible, the entire design should be a synchronous machine with an asynchronous reset. The registered part of the definition should use one clock and one reset in the sensitivity list:

```
always @(posedge pci_clk or negedge resetz)
```

7.5.6 Asynchronous Boundaries

There are 2 major clock domains in the design: pci_clk (up to 33MHz), sclk (49.152Mhz clock from the physical interface) and link_clk (24.576MHz from sclk/2). Note that sclk and link_clk are closely related, thus basically in the same clock domain.

The design should minimize the number of modules containing more than one clock.

The asynchronous boundaries are:

1. LINK block: sclk to link_clk. The sclk to link_clk boundary is limited to a very few modules located at the PHY/LINK interface. These modules produce link_clk, which is the sclk divided by 2.

2. LINK block: link_clk to pci_clk. The link_clk to pci_clk boundary in the LINK block is limited to synchronizing PCI slave register reads and writes.

The link_clk is used in the remainder of the link logic.

3. FIFO block: link_clk to pci_clk. The FIFO block is the primary asynchronous boundary between the PCI clock domain and the sclk/link_clk clock domain.

7.5.7 Synthesizable Verilog.

Use case statements where convenient, cover all cases with a default statement.

Avoid use of arithmetic's unless needed.

Keep source code as simple as possible.

Describe synthesizable hardware with code (as opposed to complex software constructs) as much as possible.

Synthesize sample code constructs to understand relation of source code to resulting synthesized gates.

For registered logic, limit process sensitivity list to the clock and the (one) asynchronous reset.

For combinatorial logic, the process sensitivity list should include only the required signals. The process sensitivity list should include all input signals; but should not include extra, unused inputs.

Simulate with synthesis checks enabled.

8. APPENDIX B - SIGNAL TO PACKAGE PIN ASSIGNMENTS

GROUND PINS = 24; 3.3 VOLT VCC PINS = 23; 5.0 VOLT REFERENCE PINS = 4; SIGNAL PINS = 121 NOT CONNECTED = 4; TOTAL PINS = 176

Signal Name	Pin #	Signal Name	Pin #	Signal Name	Pin #	Signal Name	Pin #
3.3V VCC	1	GND	45	GND	89	seeprom_clk	133
N/C (spare)	2	N/C (reserved)	46	aux_data5	90	seeprom_data	134
pci_ad25	3	pci_ad8	47	aux_data4	91	5V	135
pci_ad24	4	pci_cbez0	48	aux_data3	92	link_isoz	136
pci_cbez3	5	3.3V VCC	49	3.3V VCC	93	link_cyclein	137
GND	6	pci_ad7	50	aux_data2	94	3.3V VCC	138
pci_idsel	7	GND	51	GND	95	link_cycleout	139
3.3V VCC	8	pci_ad6	52	aux_data1	96	test_out	140
pci_ad23	9	pci_ad5	53	aux_data0	97	GND	141
pci_ad22	10	pci_ad4	54	aux_adr15	98	phy_ctl0	142
pci_ad21	11	pci_ad3	55	aux_adr14	99	phy_ctl1	143
5.0V VCC	12	3.3V VCC	56	3.3V VCC	100	phy_lreq	144
pci_ad20	13	pci_ad2	57	aux_adr13	101	3.3V VCC	145
GND	14	pci_ad1	58	GND	102	phy_data0	146
pci_ad19	15	pci_ad0	59	aux_adr12	103	phy_data1	147
pci_ad18	16	5.0V VCC	60	aux_adr11	104	phy_data2	148
pci_ad17	17	aux_intz	61	aux_adr10	105	phy_data3	149
pci_ad16	18	aux_rdy	62	aux_adr9	106	GND	150
3.3 VCC	19	5.0V VCC	63	3.3 VCC	107	phy_data4	151
pci_cbez2	20	aux_clk	64	aux_adr8	108	phy_data5	152
GND	21	GND	65	5.0V VCC	109	phy_data6	153
pci_framez	22	aux_rstz	66	aux_adr7	110	phy_data7	154
pci_irdyz	23	ram_csz	67	aux_adr6	111	GND	155
pci_trdyz	24	rom_csz	68	aux_adr5	112	phy_clk50	156
pci_devselz	25	aux_csz	69	aux_adr4	113	3.3V VCC	157
3.3V VCC	26	3.3 VCC	70	GND	114	test_enable	158
pci_stopz	27	aux_wez1	71	aux_adr3	115	auto_boot	159
GND	28	GND	72	3.3V VCC	116	GND	160

N/C (reserved)	29	aux_wez0	73	aux_adr2	117	pci_clk	161
pci_perrz	30	aux_oez	74	aux_adr1	118	5.0V VCC	162
pci_serrz	31	3.3V VCC	75	aux_adr0	119	pci_resetz	163
pci_par	32	aux_data15	76	N/C (spare)	120	pci_gntz	164
3.3V VCC	33	aux_data14	77	GND	121	3.3V VCC	165
pci_cbez1	34	aux_data13	78	gpio_data3	122	pci_intaz	166
GND	35	GND	79	gpio_data2	123	pci_reqz	167
pci_ad15	36	aux_data12	80	gpio_data1	124	GND	168
pci_ad14	37	aux_data11	81	gpio_data0	125	pci_ad31	169
pci_ad13	38	aux_data10	82	zv_pix_clk	126	pci_ad30	170
pci_ad12	39	aux_data9	83	zv_vsync	127	pci_ad29	171
5.0V VCC	40	5.0V VCC	84	3.3V VCC	128	3.3V VCC	172
pci_ad11	41	aux_data8	85	zv_ext_clk	129	pci_ad28	173
3.3V VCC	42	3.3V VCC	86	GND	130	pci_ad27	174
pci_ad10	43	aux_data7	87	zv_hsync	131	GND	175
pci_ad9	44	aux_data6	88	zv_data_valid	132	pci_ad26	176

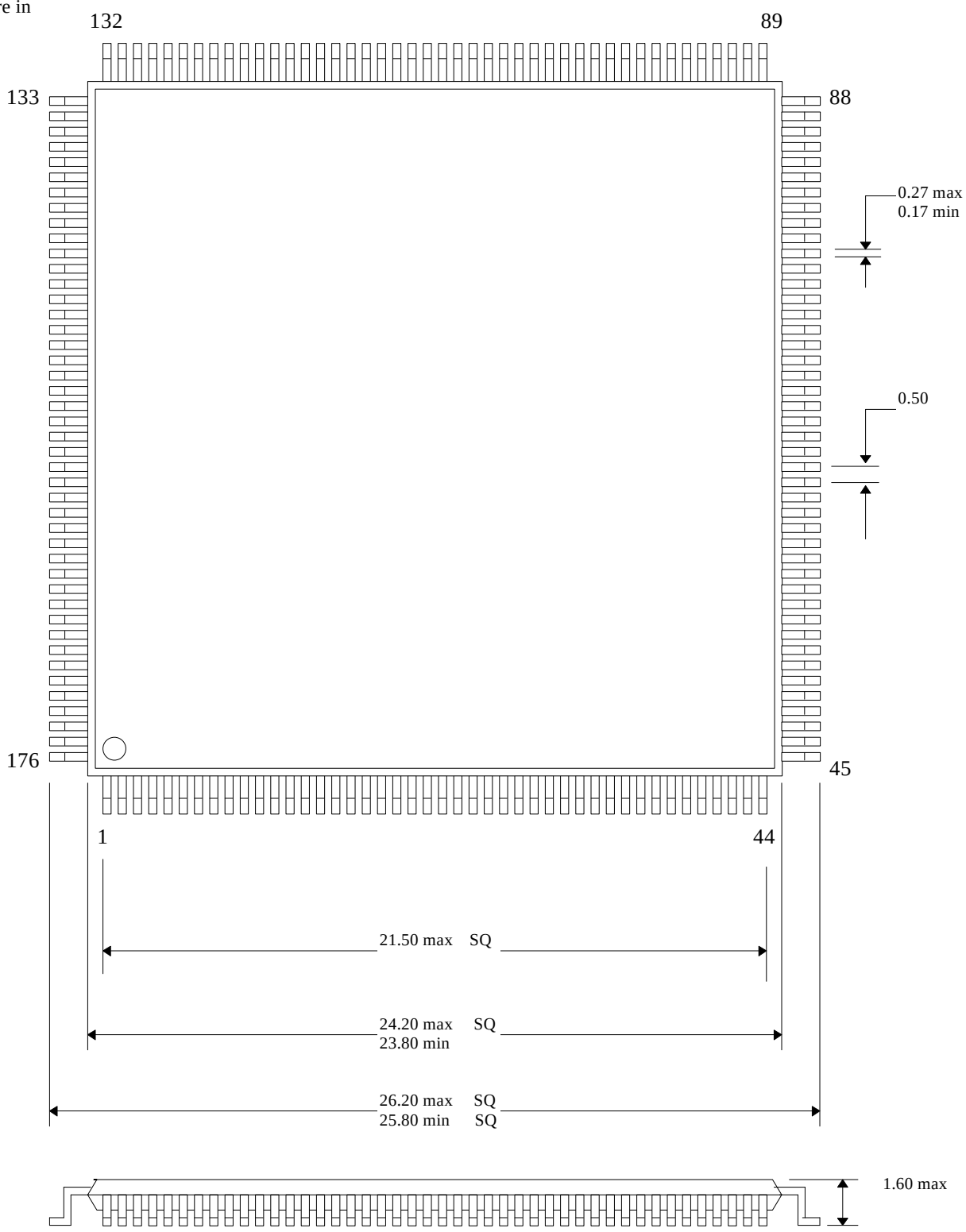
PCI-LYNX I/O SIGNAL FUNCTIONAL DESCRIPTION

Signal Name	Dir	Functional Description
GND	I	Ground
3.3V VCC	I	3.3 Volt Power
5V	I	5 Volt Tolerance Input
pci_clk	I	PCI- System clock. 0-33 MHz
pci_ad[31:0]	I/O	PCI- Multiplexed address/data bus signals
pci_cbez[3:0]	I/O	PCI- Multiplexed command/byte enable signals
pci_par	I/O	PCI- Parity signal. Parity is even across pci_ad [31:0] and pci_cbez[3:0] signals
pci_framez	I/O	PCI- frame signal
pci_irdyz	I/O	PCI- Initiator ready signal
pci_trdyz	I/O	PCI- Target ready signal
pci_devselz	I/O	PCI- Device select
pci_stopz	I/O	PCI- Stop
pci_idselz	I/O	PCI- Initialization device select
pci_perrz	I/O	PCI- Data parity error
pci_serrz	OD	PCI- System error. This is an open drain signal.
pci_reqz	O	PCI- Master bus request to PCI bus arbiter
pci_gntz	I	PCI- Bus grant from PCI bus arbiter
pci_resetz	I	PCI- System reset
pci_intaz	OD	PCI- System interrupt A. This is an open drain signal
seeprom_data	I/O	External Serial EEPROM read-write data line
seeprom_clk	I/O	External Serial EEPROM data clock
aux_clk	O	Auxiliary port clock out (output at frequency of PCI Clock)
aux_rstz	O	Auxiliary port reset out
aux_intz	I	Auxiliary port interrupt in
gpio_data[3:0]	I/O	Auxiliary port general purpose programmable i/o signals
aux_adr[15:0]	O	Auxiliary port address lines out to external logic
aux_data[15:0]	I/O	Auxiliary port bi-directional data bus to external logic
aux_oez	O	Auxiliary port output enable to enable external logic data on to the aux_data bus
aux_wrz[1:0]	O	Auxiliary port write strobes to external logic
aux_rdy	I	Auxiliary port ready indication from external logic
aux_csz	O	Auxiliary port chip select to external logic
rom_csz	O	external ROM chip select
ram_csz	O	external RAM chip select
phy_ctl[0:1]	I/O	Phy-link bi-directional control lines
phy_data[0:7]	I/O	Phy-link bi-directional data lines
phy_clk50	I	50 Mhz System clock from PHY chip.
phy_lreq	O	Phy-link request signal generated by the PCI-lynx chip
link_isoz	I	Phy-link isolation barrier mode
link_cyclein	I	Optional external 8Khz clock for use as the cycle clock.
link_cycleout	O	Cycle timer 8Khz cycle clock out
zv_ext_clk	I	Zoom port external clock input
zv_vsync	O	Zoom port vertical sync output
zv_hsync	O	Zoom port horizonal sync output
zv_data_valid	O	Zoom port data valid signal
test_out	O	Test mux out. Internal test point selected by the test multiplexer for observation.
test_enable	I	Test enable. Enables factory test features.
autoboot	I	Autoboot. Selects autoboot mode.

9. APPENDIX C - ASIC Package Outline Dimension Drawing

Figure 16. 176 pin Plastic Quad Flat Pack (S-PQFP-G176)

NOTE: All dimensions are in millimeters



10. APPENDIX D - FIFO PACKET ORGANIZATION FORMATS

ASYNCHRONOUS TRANSMIT FIFO SINGLE DATA QUADLET PACKET FORMAT

32	31					0
1	START_OF_PACKET_CONTROL_WORD					
0	DESTINATION_ID	TLABEL	RT	TCODE	PRIORITY	
0	SOURCE_ID	DESTINATION OFFSET HI				
0	DESTINATION OFFSET LOW					
0	QUADLET DATA (for write request and read response)					
1	END_OF_PACKET_CONTROL_WORD					

Field Name	Bit Positions	Description
START_OF_PACKET CONTROL_WORD	31 - 0	FIFO start control word. Marks the start of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 114
DESTINATION_ID	31 - 16	This is the concatenation of the 10-bit bus number and the 6-bit node number which forms the destination address of the node which this packet is being sent to.
TLABEL	15 - 10	This field is the transaction label, which is a unique tag for each outstanding transaction between two nodes. This is used to pair up a response packet with a corresponding request packet.
RT	9 - 8	Retry code field
TCODE	7 - 4	The transaction code for this packet. (See table 6-9 of IEEE 1394-1995 Standard)
PRIORITY	3 - 0	The priority level for this packet. For cable implementation the value of the bits must be zero.
SOURCE_ID	31 - 16	This is the concatenation of the 10-bit bus number and the 6-bit node number which forms the destination address of this packet
DESTINATION OFFSET HI DESTINATION OFFSET LOW	15 - 0 31 - 0	The concatenation, of these two fields addresses a quadlet in the destination node address space. This address must be quadlet-aligned(modulo-4)
QUADLET DATA	31 - 0	For write requests and read responses, this field holds the data to be transferred. For write responses and read requests, this field is not used and should not be written into the FIFO.
END_OF_PACKET CONTROL_WORD	31 - 0	FIFO end control word. Marks the end of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 114

ASYNCHRONOUS TRANSMIT FIFO MULTIPLE DATA QUADLET FORMAT

32	31				0
1	START_OF_PACKET_CONTROL_WORD				
0	DESTINATION_ID	TLABEL	RES	TCODE	PRIORITY
0	SOURCE_ID	DESTINATION OFFSET HI			
0	DESTINATION OFFSET LOW				
0	DATA LENGTH	EXTENDED_TCODE			
0	BLOCK DATA (for write request and read response)				
1	END_OF_PACKET_CONTROL_WORD				

Field Name	Bit Positions	Description
START_OF_PACKET CONTROL_WORD	31 - 0	FIFO start control word. Marks the start of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 114
DESTINATION_ID	31 - 16	This is the concatenation of the 10-bit bus number and the 6-bit node number which forms the destination address of the node which this packet is being sent to.
TLABEL	15 - 10	This field is the transaction label, which is a unique tag for each outstanding transaction between two nodes. This is used to pair up a response packet with a corresponding request packet.
RES	9 - 8	Reserved
TCODE	7 - 4	The transaction code for this packet. (See table 6-9 of IEEE 1394-1995 Standard)
PRIORITY	3 - 0	The priority level for this packet. For cable implementation the value of the bits must be zero.
DESTINATION ID	31 - 16	This is the concatenation of the 10-bit bus number and the 6-bit node number which forms the destination address of this packet
DESTINATION OFFSET HI DESTINATION OFFSET LOW	15 - 0 31 - 0	The concatenation, of these two fields addresses a quadlet in the destination node address space. This address must be quadlet-aligned(modulo-4)
DATA LENGTH	31 - 16	The number of bytes of data to be transmitted in the packet.
EXTENDED TCODE	15 - 0	The block extended tcode to be performed on the data in this packet. See table 6-11 of the IEEE 1394-1995
BLOCK DATA	31 - 0	The data to be sent. If data length is 0, no data should be written into the FIFO for this field. Regardless of the destination or source alignment of the data, the first byte of the block must appear in byte 0 of the first quadlet.
END_OF_PACKET CONTROL_WORD	31 - 0	FIFO end control word. Marks the end of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 114

ASYNCHRONOUS RECEIVE FIFO SINGLE DATA QUADLET FORMAT

32	31				0
1	START_OF_PACKET_CONTROL_WORD				
0	DESTINATION_ID	TLABEL	RT	TCODE	PRIORITY
0	SOURCE_ID	DESTINATION OFFSET HI			
0	DESTINATION OFFSET LOW				
0	QUADLET DATA (for write request and read response)				
1	END_OF_PACKET_CONTROL_WORD				

Field Name	Bit Positions	Description
START_OF_PACKET CONTROL_WORD	31 - 0	FIFO start control word. Marks the start of a packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 115
DESTINATION_ID	31 - 16	This is the concatenation of the 10-bit bus number and the 6-bit node number which forms the destination address of the node which this packet is being sent to.
TLABEL	15 - 10	This field is the transaction label, which is a unique tag for each outstanding transaction between two nodes. This is used to pair up a response packet with a corresponding request packet.
RT	9 - 8	The retry code for this packet. 00 = new, 10 = retryA, 11 = retryB
TCODE	7 - 4	The transaction code for this packet. (See table 6-9 of IEEE 1394-1995 Standard)
PRIORITY	3 - 0	The priority level for this packet. For cable implementation the value of the bits must be zero.
SOURCE_ID	31 - 16	This is the concatenation of the 10-bit bus number and the 6-bit node number which forms the destination address of this packet
DESTINATION OFFSET HI DESTINATION OFFSET LOW	15 - 0 31 - 0	The concatenation, of these two fields addresses a quadlet in the destination node address space. This address must be quadlet-aligned(modulo-4)
QUADLET DATA	31 - 0	For write requests and read responses, this field holds the data to be transferred. For write responses and read requests, this field is not used and should not be written into the FIFO.
END_OF_PACKET CONTROL_WORD	31 - 0	FIFO end control word. Marks the end of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 115

ASYNCHRONOUS RECEIVE FIFO MULTIPLE DATA QUADLET FORMAT

32	31				0
1	START_OF_PACKET_CONTROL_WORD				
0	DESTINATION_ID	TLABEL	RT	TCODE	PRIORITY
0	SOURCE_ID	DESTINATION OFFSET HI			
0	DESTINATION OFFSET LOW				
0	DATA_LENGTH	EXTENDED_TCODE			
0	BLOCK DATA (for write request and read response)				
1	END_OF_PACKET_CONTROL_WORD				

Field Name	Bit Positions	Description
START_OF_PACKET CONTROL_WORD	31 - 0	FIFO start control word. Marks the start of a packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 115
DESTINATION ID	31 - 16	This is the concatenation of the 10-bit bus number and the 6-bit node number which forms the destination address of the node which this packet is being sent to.
TLABEL	15 - 10	This field is the transaction label, which is a unique tag for each outstanding transaction between two nodes. This is used to pair up a response packet with a corresponding request packet.
RT	9 - 8	The retry code for this packet. 00 = new, 10 = retryA, 11 = retryB
TCODE	7 - 4	The transaction code for this packet. (See table 6-9 of IEEE 1394-1995 Standard)
PRIORITY	3 - 0	The priority level for this packet. For cable implementation the value of the bits must be zero.
SOURCE ID	31 - 16	This is the node ID of the sender of this packet.
DESTINATION OFFSET HI DESTINATION OFFSET LOW	15 - 0 31 - 0	The concatenation, of these two fields addresses a quadlet in the destination node address space. This address must be quadlet-aligned(modulo-4). The upper 4 bits of the destination offset high field are used as the response code for lock response packets.
DATA_LENGTH	31 - 16	For write requests, read responses, and locks, this field indicates the number of bytes being transferred. For read requests, this field indicates the number of bytes of data to be read. A write response packet does not use this field.
EXTENDED TCODE	15 - 0	The block extended tcode to be performed on the data in this packet. See table 6-11 of the IEEE 1394-1995 serial bus specification.
BLOCK DATA	31 - 0	This field contains any data being transferred for this packet. Regardless of the destination address or memory alignment, the first byte of the data appears in byte 0 of the first quadlet of this field. the last quadlet of this field is padded with zeros out to 4 bytes, if necessary.
END_OF_PACKET CONTROL_WORD	31 - 0	FIFO end control word. Marks the end of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 115

ISOCHRONOUS TRANSMIT FIFO PACKET FORMAT

32	31					0
1	START_OF_PACKET_CONTROL_WORD					
0	DATA LENGTH	TAG	CHANNEL_NO	TCODE	SY	
0	ISOCHRONOUS DATA					
1	END_OF_PACKET_CONTROL_WORD					

Field Name	Bit Positions	Description
START_OF_PACKET CONTROL_WORD	31 - 0	FIFO start control word. Marks the start of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 116
DATA LENGTH	31 - 16	Indicates the number of bytes in the ISO packet.
TAG	15 - 14	TBD
CHANNEL_NO	13 - 8	The channel number that this packet is being transmitted to.
TCODE	7 - 4	Transaction code = 1010
SY	3 - 0	Transaction layer specific synchronization bits
ISOCHRONOUS DATA	31 - 0	The data to be transmitted in this packet. The first byte of data must appear in byte 0 of the first quadlet of this field. If the last quadlet does not contain four bytes of data, the unused bytes should be padded with zeroes.
END_OF_PACKET CONTROL_WORD	31 - 0	FIFO end control word. Marks the end of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 116

ISOCHRONOUS RECEIVE FIFO PACKET FORMAT

32	31					0
1	START_OF_PACKET_CONTROL_WORD					
0	DATA LENGTH	TAG	CHANNEL_NO	TCODE	SY	
0	ISOCHRONOUS DATA					
1	END_OF_PACKET_CONTROL_WORD					

Field Name	Bit Positions	Description
START_OF_PACKET CONTROL_WORD	31 - 0	FIFO start control word. Marks the start of a packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 114
DATA LENGTH	31 - 16	Indicates the number of bytes in the ISO packet.
TAG	15 - 14	TBD
CHANNEL_NO	13 - 8	The channel number that this packet is being transmitted to.
TCODE	7 - 4	Transaction code = 1010
SY	3 - 0	Transaction layer specific synchronization bits
ISOCHRONOUS DATA		The data to be transmitted in this packet. The first byte of data must appear in byte 0 of the first quadlet of this field. If the last quadlet does not contain four bytes of data, the unused bytes should be padded with zeroes.
END_OF_PACKET CONTROL_WORD	31 - 0	FIFO end control word. Marks the end of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in APPENDIX E on page 114

11. APPENDIX E - FIFO CONTROL WORD AND TRANSMIT ACK FORMATS

General Receive FIFO Isochronous Packet Control Token Format Definition

32	31	30 27	26	25 24	23 18	17	16	15 13	12 0
FCW	PKTBD	RCV_STAT	RES	RCV_SPD	DMA_CH	ISO	SELF_ID	RES	PACKET_SIZE

Bit Field	Function Description
FCW	Control word specified. FCW = 1 indicates that bits 31 - 0 of the quadlet are to be interpreted as a FIFO packet control word. The bit fields defined for this control word shall only be valid when FCW = 1.
PKTBD	Packet delimiter. PKTBD = 0 indicates start of packet PKTBD = 1 indicates end of packet
RCV_STAT	Packet receive status. This field is valid when PKTBD = 1 0001 - Packet was successfully received. If the packet was a request subaction, the destination node has successfully completed the transaction and no response subaction shall follow. 1101 - The receiver could not accept the packet because a CRC error occurred or the length of the data block payload did not match the length contained in the data length field of the packet header
RCV_SPEED	The speed at which the packet was received. 00 = 100 mbps 01 = 200 mbps. This field valid when PKTBD= 0 and 1
DMA_CH	The DMA channel number assigned to the packet. The value of this number shall be from 000000 to 111111. This field valid when PKTBD = 0 and 1
ISO	ISO = 1 Isochronous Packet Type
SELF_ID	SELF_ID = 1 indicates that this packet is a self_id packet
PACKET_SIZE	The total size of the packet in bytes (header + data payload). This field valid when PKTBD = 0 and 1

General Receive FIFO Asynchronous Packet Control Token Format Definition

32	31	30 27	26	25 24	23 18	17	16	15 13	12 0
FCW	PKTBD	ACK_SENT	RES	RCV_SPD	DMA_CH	ISO	SELF_ID	RES	PACKET_SIZE

Bit Field	Function Description
FCW	Control word specified. FCW = 1 indicates that bits 31 - 0 of the quadlet are to be interpreted as a FIFO packet control word. The bit fields defined for this control word shall only be valid when FCW = 1.
PKTBD	Packet delimiter. PKTBD = 0 indicates start of packet PKTBD = 1 indicates end of packet
ACK_SENT	Packet receive status. This field is only valid when PKTBD = 1. 0001 - Packet was successfully received. If the packet was a request subaction, the destination node has successfully completed the transaction and no response subaction shall follow. 0010 - Ack Pending. Packet was successfully received. If the Packet was a request subaction, a subaction response will follow at a later time. 0100 - The packet could not be accepted. The destination transaction layer may accept the packet on a retry X of the subaction. 0101 - The packet could not be accepted. The destination transaction layer will accept the packet when the node is not busy during the next occurrence of retry phase A. 0110 - The packet could not be accepted. The destination transaction layer will accept the packet when the node is not busy during the next occurrence of retry phase B. 1101 - The receiver could not accept the packet because a CRC error occurred or the length of the data block payload did not match the length contained in the data length field of the packet header. 1110 - A field in the request packet header was set to an unsupported or incorrect value, or an invalid transaction was attempted.
RCV_SPD	The speed at which the packet is received at. 00 = 100 mbps 01 = 200 mbps. This field valid for PKTBD = 0 and 1
DMA_CHANNEL	The DMA channel number assigned to the packet. The value of this number shall be from 000000 to 111111. This field is valid when PKTBD = 0 and 1
ISO	ISO = 0 Asynchronous Packet Type
SELF_ID	SELF_ID = 1 This packet is a self id packet type
PACKET_SIZE	The total size of the packet in bytes (header + data payload). This field valid for PKTBD = 0 and 1

Isochronous Transmit FIFO Control Word Format

32	31 30	29 28	27	26	25	24 - 0
FCW	PKTBNDRY	SPD_CODE	MSTR_ERR	RESERVED	UNFORMATTED XMT	RESERVED

Bit Field	Function Description
FCW	Control word specified. FCW = 1 indicates that bits 31 - 0 of the quadlet are to be interpreted as a FIFO packet control word. The bit fields defined for this control word shall only be valid when FCW = 1.
PKTBNDRY	Packet delimiter. PKTBNDRY = 00 indicates start of packet PKTBNDRY = 10 indicates end of packet PKTBNDRY = 11 indicates end of packet and the last packet to be transmitted for the current isochronous interval.
SPD_CODE	Transmit speed code. SPD_CODE = 00 - 100mbps SPD_CODE = 01 - 200mbps This field is valid for PKTBNDRY = 00
MSTR_ERR	Master Error. Indicates if an error occurred during the transfer of the packet from host memory to the Asynchronous transmit FIFO. Error occurred if set to 1. No error occurred if set to 0
UNFORMATTED XMT	When set to logic 1, the transmitter shall transmit the data quadlets between the packet start and end control tokens without performing the normal packet formatting checks and header-data CRC insertions.

Asynchronous Transmit FIFO Control Word Format

32	31 30	29 28	27	26	25	24 - 0
FCW	PKTBNDRY	SPD_CODE	MSTR_ERR	RT	UNFORMATTED XMT	RESERVED

Bit Field	Function Description
FCW	Control word specified. FCW = 1 indicates that bits 31 - 0 of the quadlet are to be interpreted as a FIFO packet control word. The bit fields defined for this control word shall only be valid when FCW = 1.
PKTBNDRY	Packet delimiter. PKTBNDRY = 00 indicates start of packet PKTBNDRY = 10 indicates end of packet
SPD_CODE	Transmit speed code. SPD_CODE = 00 - 100mbps SPD_CODE = 01 - 200mbps This field is valid for PKTBNDRY = 00
RT	Transmit Packet Retry
MSTR_ERR	Master Error. Indicates if an error occurred during the transfer of the packet from host memory to the Asynchronous transmit FIFO. Error occurred if set to 1. No error occurred if set to 0
UNFORMATTED XMT	When set to logic 1, the transmitter shall transmit the data quadlets between the packet start and end control tokens without performing the normal packet formatting checks and header-data CRC insertions.

Asynchronous Transmit Acknowledge Codes Returned to DMA channel After an Asynchronous packet Transmission completes.

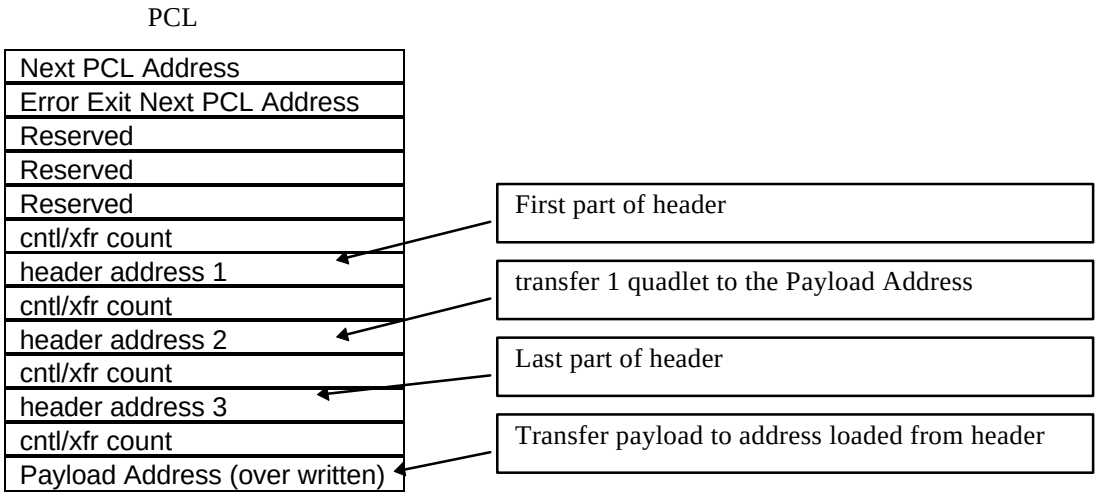
Acknowledge codes Returned To Active Transmit DMA channel					Functional Description
ack3	ack2	ack1	ack0	ack_type	
0	0	0	1	0	**Ack_Complete**- packet was successfully transmitted
0	0	1	0	0	**Ack_Pending**- packet successfully transmitted but a response transaction will follow at a later time
0	1	0	0	0	**Ack_busy_X Received**- The receiving node could not accept the packet. Retry the packet transmission using BUSY_X retry code.
0	1	0	1	0	**Ack_busy_A Received**- The receiving node could not accept the packet. Retry the packet transmission using BUSY_X retry code.
0	1	1	0	0	**Ack busy_B Received**- The receiving node could not accept the packet. Retry the packet transmission using BUSY_X retry code.
1	1	0	1	0	**Ack Data Error Received**- The receiving node could not accept the block packet because the data field CRC check failed, the number of data bytes received did not match the data byte count of the packet.
1	1	1	0	0	**Ack Type Error Received**- The receiving node detected a field in the request packet header was set to an unsupported or incorrect value, or an invalid transaction was attempted (e.g., a write to a read-only address).
0	0	0	0	1	**Retry time out**- The current ASYNC packet transmission retry count has timed out without a successful transmission occurring.
0	0	0	1	1	**No Ack Received**- An ack was expected but not received within the 1394 gap time allowed.
0	0	1	0	1	**Transmit FIFO Underrun**- The entire packet was not transmitted because the PCI bus was not able to keep up with the 1394 bus.
1	1	1	0	1	**Improper Packet Format**- Packet was not transmitted because of a malformed header.

12. APPENDIX F - Program Control List (PCL) Examples

12.1 APPENDIX F.1 - TRANSFER "AT ADDRESS" PROGRAM

The LYNX PCL can be coded such that the payload data is transferred to the host address contained in the 1394 packet header address field. This is accomplished by simply transferring the header quadlet containing the address into its own PCL such to modify the data buffer address for the payload data.

For example:



12.2 APPENDIX F.2 - TRANSFER "CONTIGUOUS VIRTUAL MEMORY" PROGRAM

13. APPENDIX G - SERIAL EEPROM ADDRESS MAP

The PCI LYNX loads certain internal configuration registers from serial EEPROM immediately after power reset. During the time the registers are being loaded, any PCI slave access to the PCI LYNX will be terminate with a retry disconnect. This ensures that the system software will always read the values loaded from Serial EEPROM whenever the PCI LYNX is first accessed.

The first 8 bytes of the Serial EEPROM address space are reserved for use by the PCI LYNX after power reset. These bytes are loaded into internal registers by the PCI LYNX are described in the following table. All remaining bytes in the Serial EEPROM are available for software to read and write via the Serial EEPROM Control Register. These bytes might be used for information such as 1394 unique ID, assembly part number, manufacture, assembly revision information, manufacturing data, etc. The size of the Serial EEPROM address space depends on which serial EEPROM device is selected to be used with the PCI LYNX.

Table 1 - Serial EEPROM Address Map

Byte Adr	Byte Description
00	PCI max_lat (Configuration Reg 3F)
01	PCI min_gnt (Configuration Reg 3E)
02	Local Bus Control Register - ROM Control (Configuration Reg B0)
03	PCI SubSystem Vendor ID (lsbyte) (Configuration Reg 2C)
04	PCI SubSystem Vendor ID (msbyte) (Configuration Reg 2D)
05	PCI SubSystem ID (lsbyte) (Configuration Reg 2E)
06	PCI SubSystem ID (ms byte) (Configuration Reg 2F)
07	Checksum (bytes 0-6)
08	User Defined
09	.
10	.
.	.
.	.
255	User Defined