PCILynx 1394 to PCI Bus Interface
TSB12LV21BPGF
Functional Specification

SCPA020A
April 1999

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Contents

List of Figures

List of Tables

PCILynx 1394 to PCI Bus Interface TSB12LV21BPGF

1 Introduction

1.1 Scope of Document

This document describes the application specific integrated circuit (ASIC) which uses Texas Instruments Inc TGC3000T ASIC technology. This device performs 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.

1.2 Feature Set

The PCILynx device supports the following features:

- Compliant with IEEE 1394–1995
- Compliant with PCI Bus Specification Revision 2.1
- Performs function of 1394 cycle master
- Detects lost cycle start messages
- Generates 32-bit CRC for transmission of 1394 packets
- Performs 32-bit CRC checking on reception of 1394 packets
- Supports IEEE 1394 transfer rates of 100, 200 and 400 Mbps
- Provides 3 size-programmable FIFOs (asynchronous transmit +isochronous transmit + general receive)
- 4K of FIFO memory
- Includes 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
- Supports DMA transfers between 1394 and local bus RAM, ROM, AUX, or ZV
- 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 8-/16-bit 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

- Operates from 3.3-V power while maintaining 5-V tolerant inputs

1.3 Applicable Documents

The following documents apply to the PCILynx ASIC.

- PCI Bus Specification Revision 2.1
- IEEE STD 1394–1995 High Performance Serial Bus
- IEEE STD 1212–1991, IEEE Standard Control and Status Register (CSR) Architecture for Microcomputer Buses
- Texas Instruments Incorporated TGC3000T ASIC Design Manual

2 Performance Requirements

The following are the PCILynx device performance requirements:

- 1394 serial data transfer rates – 100 Mbps, 200 Mbps, 400 Mbps
- PHY Link interface clock frequency – 50 MHz
- PCI interface clock rate – 0 MHz to 33 MHz

The PCILynx requires a high performance PCI bus environment to ensure minimum FIFO overrun or underrun conditions. The absolute maximum isolated latency occurrence required to avoid FIFO over/under runs depends on several things, including the PCILynx FIFO size settings, the cacheline size, the number and size of separate data buffers, the number of any auxiliary commands, packet sizes, and the access latencies once the PCILynx has acquired PCI bus ownership.

Once the PCILynx has acquired the PCI bus, a 1394 data transfer is available at every clock. The PCILynx is capable of transferring the maximum transfer rate for the standard PCI bus (33 MHz).

In addition to data transfers, the PCILynx acquires and stores control information for every packet.

3 Mechanical Requirements

3.1 Packaging Requirements

The PCILynx ASIC is a 176-pin plastic quad flat pack (PQFP) package. The outline dimensions for this package are provided in Appendix B – *ASIC Package Outline Dimension Drawing* on page B–1.

3.2 Pin Assignment Requirements

The PCILynx ASIC implements the signal-to-pin assignments as shown in Appendix A – *Signal to Package Pin Assignments* on page A–1.

4 Hardware Functional Description

4.1 System Overview

The following diagram provides a system overview of the PCILynx ASIC as it appears in a typical system configuration.

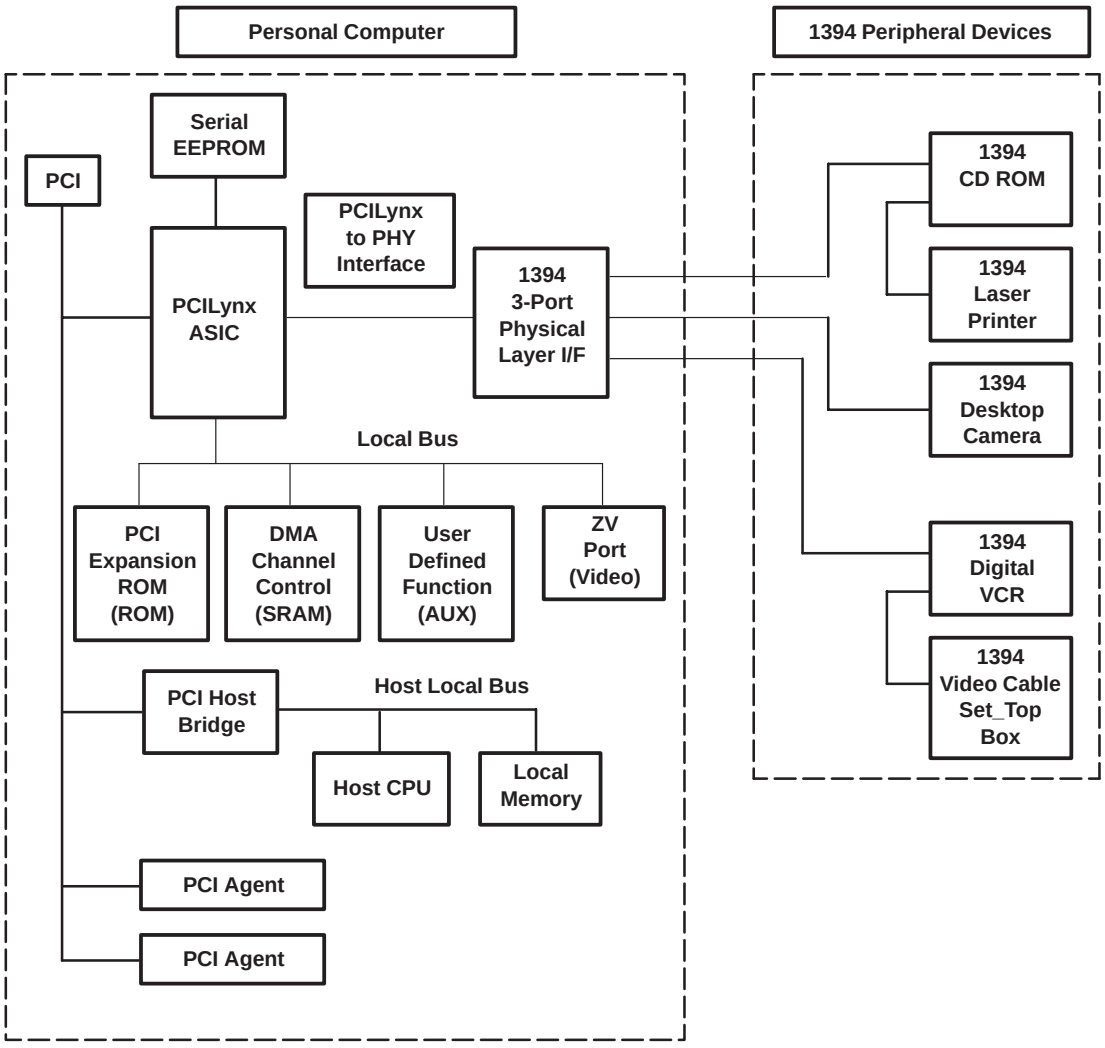


Figure 1. PCILynx ASIC in a Typical System Configuration

4.2 ASIC Functional Partitioning

The following is a block diagram that shows the functional partitioning of the PCILynx ASIC.

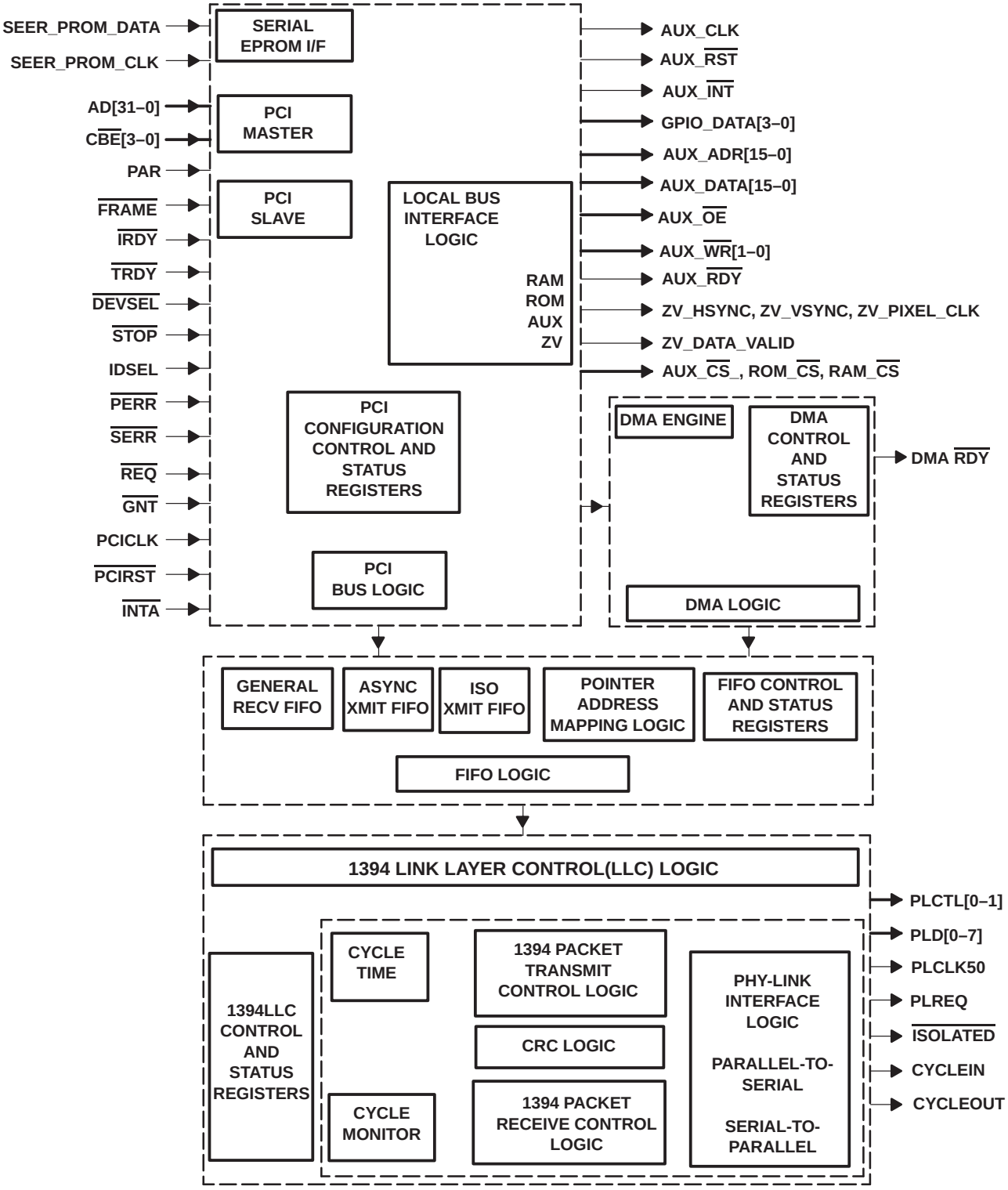


Figure 2. PCILynx Functional Partitioning

4.2.1 PCI Bus Logic

This functional block implements the logic required to interface the PCILynx ASIC to the PCI bus. The PCI bus logic meets the requirements of the PCI Bus Specification (Revision 2.1). The functional partitioning of the PCI bus logic is as follows:

- Read and write slave interface control logic for accessing all of the PCILynx control and status registers, which are required by application software to control the operation of the PCILynx and monitor its 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 PCILynx. This includes the PCI required control and base registers as well as PCILynx 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.

4.2.1.1 PCI Bus Specification (Revision 2.1) Compliance

The PCILynx complies with PCI Bus Specification (Revision 2.1). Some features of the PCILynx require software to configure options and require external hardware to meet certain criteria:

1. To meet the slave latency requirements, the user must design any external logic on the PCILynx local bus so that any PCI slave access completes in less than 16 PCI clocks. This can be accomplished by ensuring that the waitstate field for any local bus address is equal to or less than 3 for 16-bit accesses or equal to or less than 1 for 8-bit accesses. Thus, any external local bus access from the PCI bus should complete in less than 8 clocks on the local bus.

2. Software must set ENA_SLV_BURST = 0 in the miscellaneous control register (PCI configuration space offset 0x040 or PCILynx registers memory space offset 0x040).

4.2.1.2 PCI Master Logic

This logic function implements the control logic required for the PCILynx to operate on the PCI bus as a master device. This logic function meets the functional requirements for a PCI bus master device as specified in the PCI Bus Specification (Revision 2.1). As bus master, the following PCI bus commands are supported:

PCI BUS OPERATION	CMD[3–0]	PCILynx 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

4.2.1.3 PCI Slave Logic

This function implements the control logic required for the PCILynx device to operate on the PCI bus as a slave device. The logic for this function meets the functional requirements for a PCI slave device as specified in PCI Bus Specification (Revision 2.1). As a slave device, the PCILynx does not decode the I/O read and write commands. The following commands are supported:

PCI BUS OPERATION	CMD[3–0]	PCILynx SLAVE FUNCTIONS
Memory read	0110	Memory read of PCILynx addressed resource
Memory write	0111	Memory write to PCILynx addressed resource
Configuration read	1010	Configuration read of PCILynx addressed resource
Configuration write	1011	Configuration write to PCILynx addressed resource
Memory read multiple	1100	Memory read multiple to PCILynx addressed resource
Memory read line	1110	Memory read line to PCILynx addressed resource
Memory write line and invalidate	1111	Memory write to PCILynx addressed resource

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

In Revision A and later PCILynx devices, the ENA_POST_WR bit in the miscellaneous control register must be set to 0.

4.2.1.3.1 PCI Slave Address Space

The PCILynx implements the PCI configuration space as required by the PCI Bus Specification (Revision 2.1) and 4 PCI memory spaces. These PCI memory slave address spaces are specified by the base address registers contained in the PCI configuration space as shown in Figure 3.

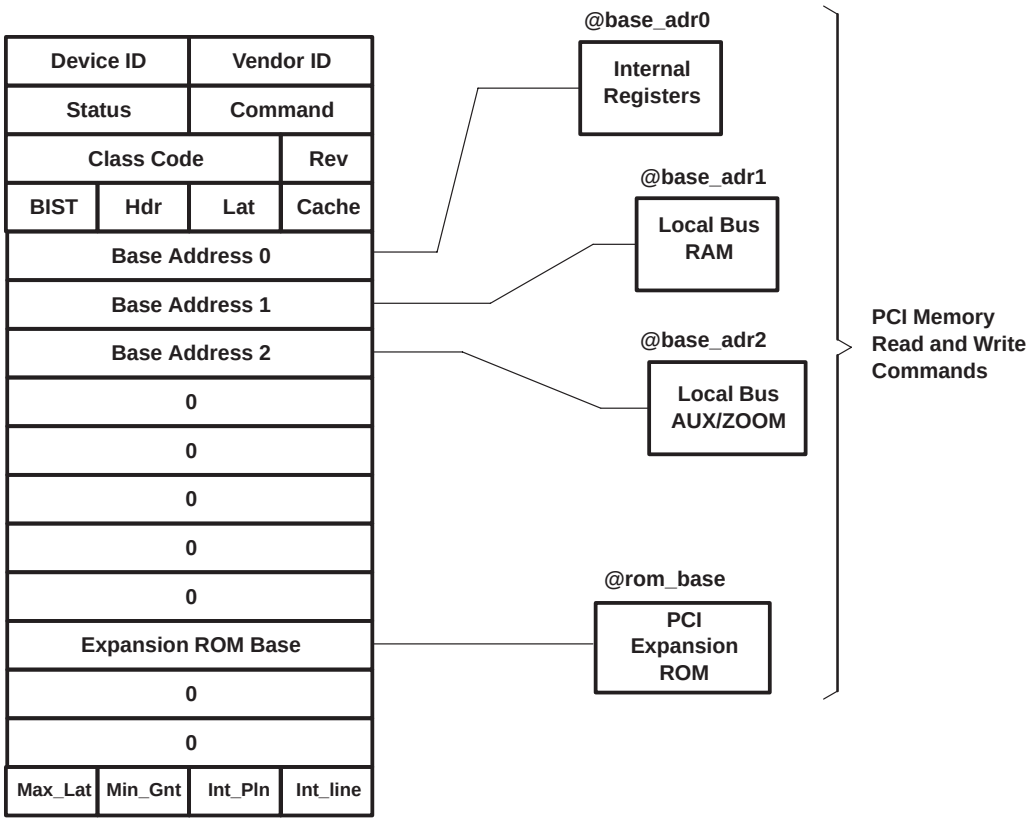


Figure 3. PCI Configuration Space for PCILynx

4.2.1.3.2 PCI Configuration Control and Status Registers

The PCILynx PCI configuration register set is defined starting in section 5.2, *PCI Configuration and Miscellaneous Register Definitions*. These registers provide system and application software with the capability to program the PCI operational configuration of the PCILynx. The functional behavior of these registers conform to the requirements of the PCI Bus Specification.

4.2.1.4 Serial EEPROM Interface

The serial EEPROM interface provides communication between PCILynx and an attached serial EEPROM. The serial EEPROM resides on an industry standard 2-wire serial bus at slave address 0. PCILynx is designed to be the only master on the 2-wire serial bus and therefore does not perform arbitration.

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:

- PCI subsystem ID
- PCI subsystem vendor ID
- PCI maximum latency
- PCI minimum grant
- ROM control

This serial EEPROM can also contain configuration data required by the 1394 command status registers as specified in *Appendix F – Serial EEPROM* 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. Reference *Phillips I^2C Peripherals Manual*.

The 5-µs timer bit in the control register provides a timing reference for timing the 2-wire serial bus protocol events, if a more accurate source is not available. 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.

4.2.1.4.1 PCI Subsystem ID and Subsystem Vendor ID Registers

Alternately, the subsystem access register at PCI offset 58h can be used to load values into the subsystem ID and subsystem vendor ID registers at PCI offset 2Ch. These registers at 2Ch are required by PC97 and later specifications to be read-only. Using the subsystem access register at offset 58h, system BIOS can write values that will be reflected back to the subsystem ID and subsystem vendor ID registers at offset 2Ch. This method is useful when no EEPROM is implemented.

4.2.1.5 Local Bus Interface Logic

The PCILynx 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 allow the PCILynx to be connected to external devices or interfaces to provide for autonomous data transfers to/from such devices.

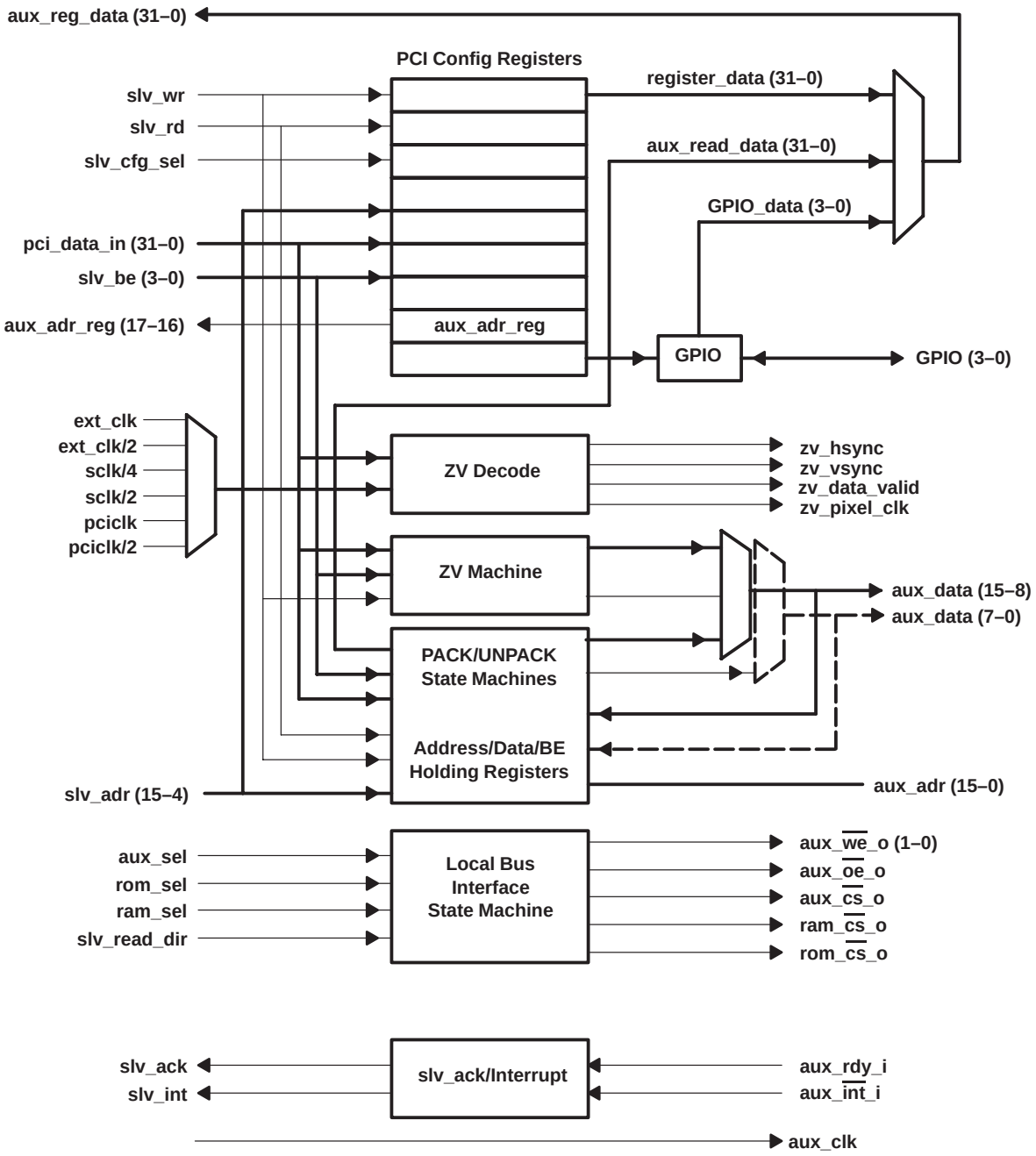


Figure 4. Local Bus Interface Block Diagram

All local bus interfaces, except the zoom video port (ZV port), are synchronous to the AUX_CLK (a buffered version of PCI clock). The ZV port clock (ZV_PIX_CLK) is programmed to be based on versions of the PCI clock, 1394 clock, or an external clock (ZV_EXT_CLK).

The local bus provides the following I/O ports:

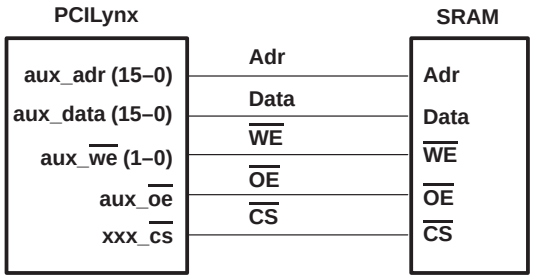
- PCI Expansion ROM
  - 64 Kbytes of address space
  - A 16-bit address bus and an 8-bit or 16-bit read or write data bus
  - Byte addressable/writable
  - Programmable wait-states/ready
  - (ROM control parameters are initialized on power up from the serial EEPROM)
- RAM
  - 64 Kbytes of address space
  - A 16-bit address bus and an 8-bit or 16-bit read or write data bus
  - Byte addressable/writable
  - Programmable wait states/ready
- AUX
  - 64 Kbytes of address space
  - A 16-bit address bus and an 8-bit or 16-bit read or write data bus
  - Byte addressable/writable
  - Programmable wait states/ready
- ZV output port
  - Sync outputs (hsync and vsync)
  - Data valid indicator
  - 8-bit or 16-bit interface
  - 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 is programmable via control registers specified in section 5.2.16, *Local Bus Control Register @0B0 {ROM, RAM, AUX, and ZV registers}*.

4.2.1.5.1 Local Bus RAM, ROM, AUX

The RAM, ROM, and AUX interfaces on the local bus all behave similarly. These interfaces are designed to allow common asynchronous RAM or ROM devices to be simply interfaced to the PCILynx. For simple designs, the RAM or ROM may be directly wired to the appropriate chip select, output enable, write enable, address bus, and data bus. In this fashion, 64 Kbytes of RAM or ROM may be directly addressed for each address space from the PCILynx.

Typical connection of a RAM device would be:



4.2.1.5.2 PCI Expansion ROM Interface

The PCI expansion ROM provides the host system with the capability of reading configuration data or executable code from an attached 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 PCI Expansion ROM access. This interface will support PCI slave and internal DMA machine read/write access to devices such as EEPROM, FLASH, and other ROM/RAM-like devices.

ROM access is controlled by the standard PCI configuration PCI expansion 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. See section 5.2.16, *Local Bus Control Register @0B0 {ROM, RAM, AUX, and ZV registers}.*

PCI EXPANSION ROM CONTROL REGISTER [7–0]		
BIT NO.	**BIT NAME**	**DESCRIPTION**
07–04	ROM_WS[3–0]	Number of wait states, 1111 = Pace transfer based on AUX_RDY with timeout
02–03	reserved	Return 0s when read.
01	ROM_WR_EN	Write enable (writable nonvolatile memory)
00	ROM_16	Data width, 1 = 16-bit data 0 = 8-bit data

4.2.1.5.3 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. See section 5.2.16, *Local Bus Control Register @0B0 {ROM, RAM, AUX, and ZV registers}*.

RAM CONTROL REGISTER [15–8]		
BIT NO.	**BIT NAME**	**DESCRIPTION**
15–12	RAM_WS[3–0]	Number of wait states, 1111 = Pace transfer based on AUX_RDY with timeout
11–09	reserved	Return 0s when read.
08	RAM_16	Data width, 1 = 16-bit data 0 = 8-bit data

4.2.1.5.4 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, then 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. See section 5.2.16, *Local Bus Control Register @0B0 {ROM, RAM, AUX, and ZV registers}*.

AUX CONTROL REGISTER [23–16]		
BIT NO.	**BIT NAME**	**DESCRIPTION**
23–20	AUX_WS[3–0]	Number of wait states, 1111 = Pace transfer based on AUX_RDY with timeout
19	INVERT_ZV_CLK	ZV clock polarity 1 = invert 0 = don’t invert
18	AUX_INT_POL	Interrupt polarity 1 = invert 0 = don’t invert
17	AUX_RSTZ	AUX port reset output (active low)
16	AUX_16	Data width, 1 = 16-bit data 0 = 8-bit data

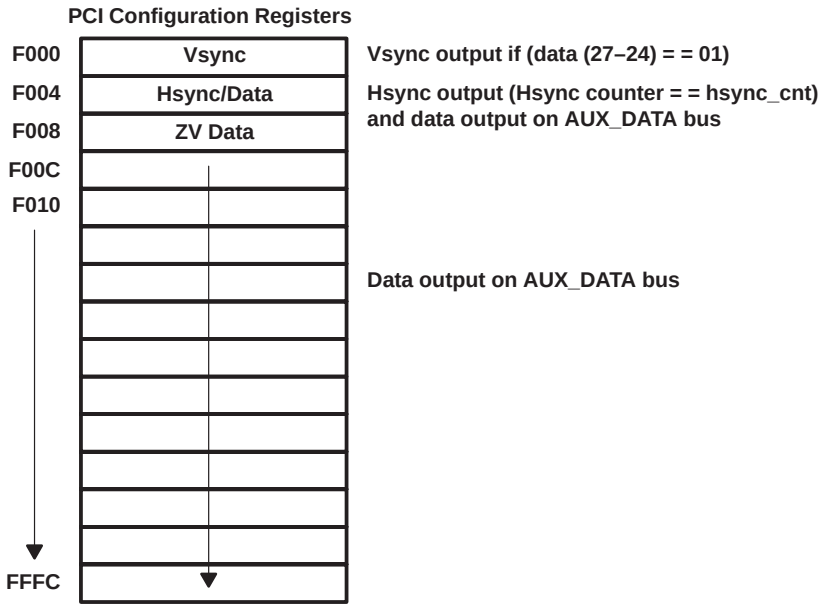
4.2.1.5.5 ZV Interface

The zoom video (ZV) port is an output-only port designed to transfer data from 1394 video devices to an external device on the PCILynx 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, then ZV is disabled and AUX claims the entire address space. When the ZV port is disabled, all ZV-related outputs are high impedance with the exception of the data bus which will still be driven during AUX, RAM, and ROM accesses.

A vertical sync output (ZV_VSYNC) is generated when a 1394 header quadlet is written to AUX address 0xF000 with the header sync field equal to 0x1 (little endian bits 27–24). The quadlet written to this address (0xF000) is not output on the AUX_DATA bus.

A horizontal sync output (ZV_HSYNC) is generated when a data quadlet is written to AUX address 0xF004 and the horizontal sync count is reached. Whether or not a horizontal sync output is generated, the data quadlet written to AUX address 0xF004 will be output on the AUX_DATA bus with the appropriate control signals (i.e., ZV_DATA_VALID and ZV_PIX_CLK).



Data quadlets transferred to all other zoom port addresses (i.e., AUX addresses between 0xF008 and 0xFFFC) are output on the AUX_DATA bus.

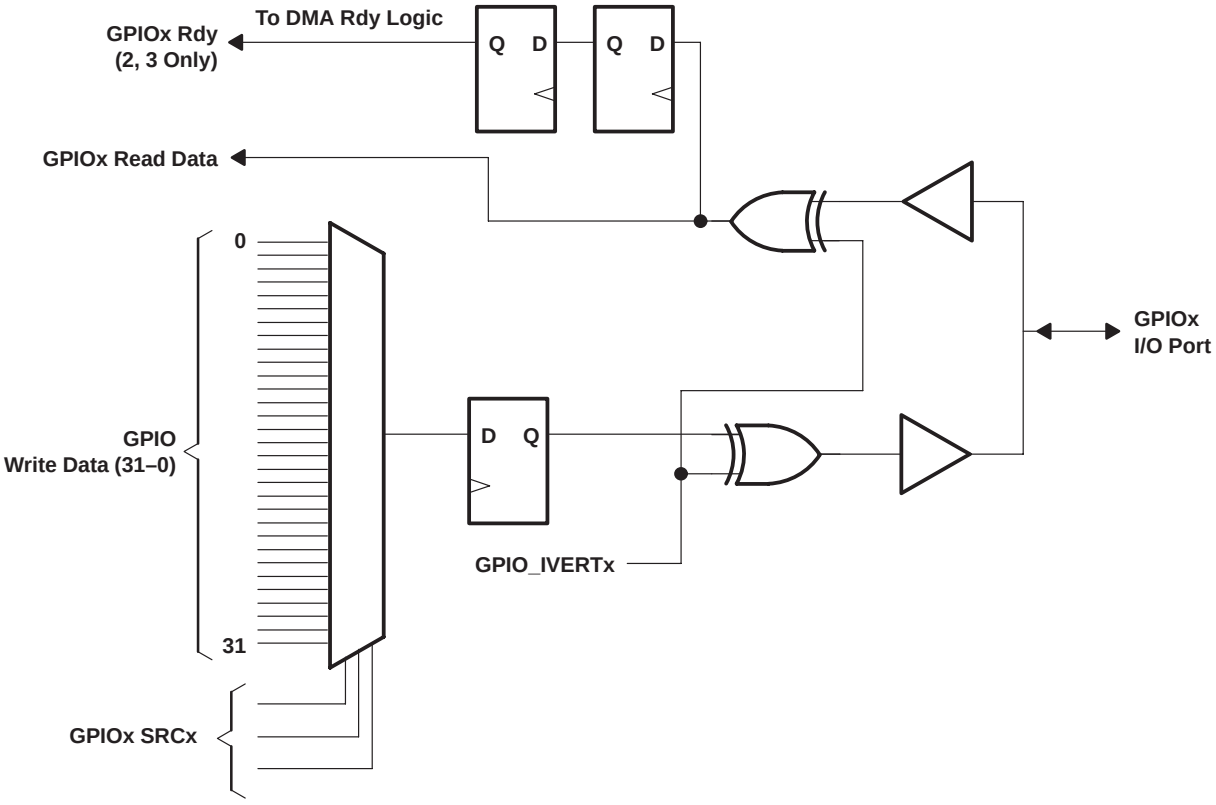
1394 data output to the ZV port must be transferred in little endian mode for the vertical sync field to be properly decoded, and for the data to be transferred in the correct sequence.

By programming the PCLs to receive 1394 digital camera packets to AUX address 0xF000, the header quadlet sync field will be evaluated to generate a vertical sync, the first quadlet of the payload will conditionally generate a horizontal sync, and the payload data will be transferred to the ZV port. See section 5.2.16, *Local Bus Control Register @0B0 {ROM, RAM, AUX, and ZV registers}*.

ZV CONTROL REGISTER [31–24]		
BIT NO.	**BIT NAME**	**DESCRIPTION**
31	GATE_PIXEL_CLK	ZV pixel clock gating enable. 0 = free running pixel clock; zv_data_valid signal must be used to determine when valid ZV data is present. 1 = gating enabled (gated mode); zv_pix_clk only toggles when valid ZV data is present.
30–28	HSYNC_CNT[2–0]	Horizontal sync count (HSYNC_CNT = 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	Data width, 1 = 16-bit data 0 = 8-bit data

4.2.1.5.6 GPIO Interface

The general-purpose I/O (GPIO) port consists of 4 general-purpose input/output ports. The operating modes of these 4 ports are independent and fully software programmable via two 32-bit control registers (16 bits per GPIO port). See section 5.2.16, *Local Bus Control Register @0B0 {ROM, RAM, AUX, and ZV registers}*.



GPIO port 0's control register bit definition is shown below.

GPIO[x] CONTROL REGISTER		
	BIT NAME	**DESCRIPTION**
	GPIO_[x]SRC[4–0]	Data bit mux select for output on GPIO[x]
	GPIO_POL[x]	Input and output polarity control (0 = noninverted, 1 = inverted)
	GIPI_OUT_EN[x]	Output enable control (0 = 3-state, 1 = enabled)

4.2.1.6 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 PCILynx 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 PCI expansion ROM is enabled, with its base address set to 0x00000000, register reads 0x00000001.
3. After power reset, DMA master access to internal PCILynx registers is enabled, with its internal register base address set to 0x00010000, register reads 0x00010001.
4. Once enabled as master on the PCI bus, the PCILynx 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 2 MS address bits are always 0. Internally, these 2 bits are used to select the PCI command.

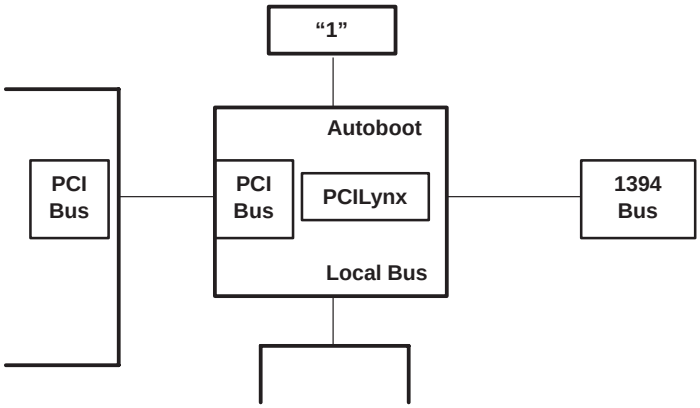
AUTOBOOT = 1		
INTERNAL PCI ADDRESS		**FUNCTION**
adr[31]	**adr[30]**	
0	X	PCI memory command
1	0	PCI I/O command
1	1	PCI configuration command

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 PCILynx can perform as the local processor to set up all the internal PCILynx 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.

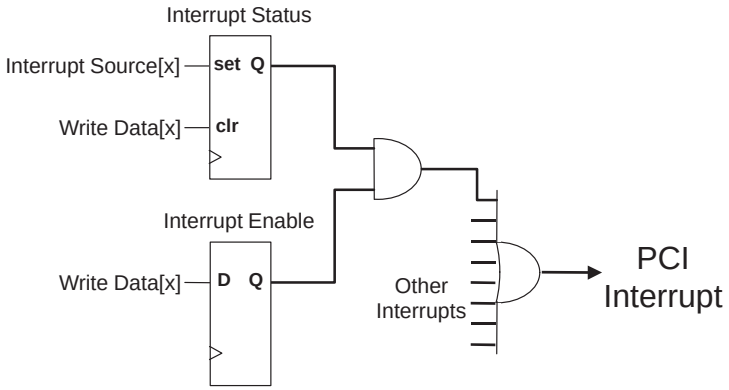
By adding external SRAM to the PCILynx, PCI slave memory is provided for devices on the PCI bus to obtain control information and have local memory for data transfers. PCL programs can then transfer device control/data via 1394 to another system.

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

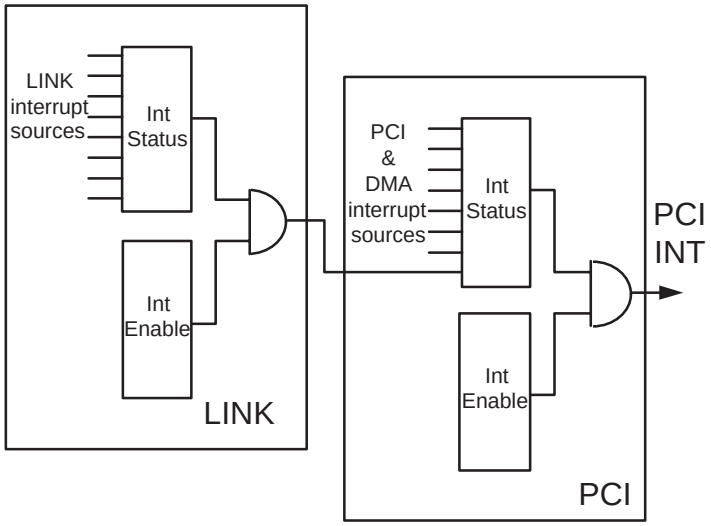


4.2.1.7 Interrupt Logic

The interrupt logic provides control for interrupts to set the PCI bus interrupt signal $\overline{\text{INT}}$ 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 PCILynx 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 status bit is set in either the PCI or LLC interrupt status registers, the status bit will set an interrupt if 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 PCILynx 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 0.

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 interrupt condition. This feature may be useful to check out interrupt software.

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

4.2.1.8 Byte Ordering (endian)

Access to the local bus resources and all PCILynx registers from the PCI bus is always in little endian (PCI bus) byte ordering. This means the least significant byte of a quadlet aligned quadlet is byte 0, and the most significant byte is byte 3.

The PCILynx is designed to accommodate 1394 transfers of either big or little endian byte ordering on the PCI bus or the local bus. The control of this ordering is specified by DMA control structures on a memory buffer basic using the big endian control bit (see the DMA section of this spec). Only transfers to or from the 1394 bus are controlled by this mechanism.

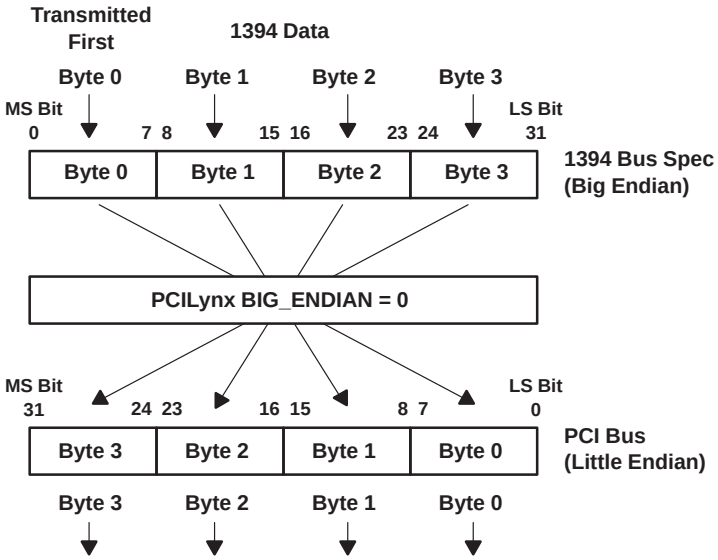


Figure 5. 1394 Data Byte Addressing is Preserved on the PCI Bus

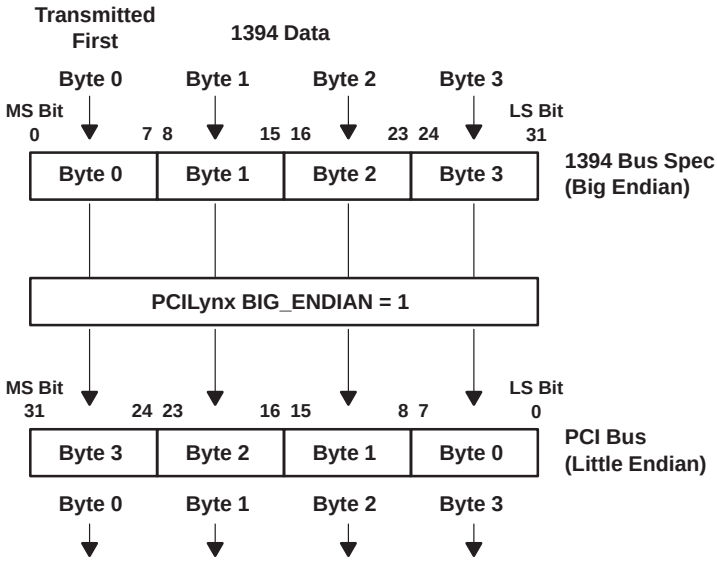


Figure 6. 1394 Data Byte Addressing not Preserved on the PCI Bus

4.2.2 DMA Logic

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

- DMA engine contains a common state machine which is priority-time multiplexed over 5 DMA channels. This block also contains 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 stand-alone processor which can build PCLs during an autoboot sequence (refer to section 4.2.1.6, *Autoboot Mode Option*). The entire scope of this functionality is not regimented and further uses will evolve over time as programming ensues.

4.2.3 FIFO Overrun and Underrun

An *underrun* is a condition where, due to PCI latencies, the transmit FIFO runs out of PCI data (FIFO is empty) while sending data to the 1394 bus. The end result is a truncated packet. For asynchronous transmits this will result in the receiver getting a CRC error and reporting an ack_data_error back to the transmitter. The software will see the data error in the PCL status and may retry it.

For isochronous (ISO) transmits underrun is a condition where there is no ack status back to the transmitter so the packet is lost. The receiver software sees the data error in the receive PCL so it knows the packet is corrupted and will probably ignore the packet.

An *overrun* is a condition where the receive FIFO gets filled up with 1394 data (FIFO is full). This can happen due to PCI latencies or because there is no PCL program running for the incoming packet. The filter registers bind the incoming packets to a particular DMA channel and hopefully software has started a PCL program for it. For asynchronous packets this will cause an ack busy to be returned to the transmitter. The transmitter can then re-send the packet. For isochronous transmission packets are just dropped because there is no return status.

The PCILynx has a test register that logs overruns and underruns. They are at offset F24. Reading these provides pretty good insight to PCI performance. If they continue to increment, especially underruns, then PCI performance is suspect, the FIFOs have been programmed too small, or the transmit threshold registers have been programmed too small for the target PCI environment.

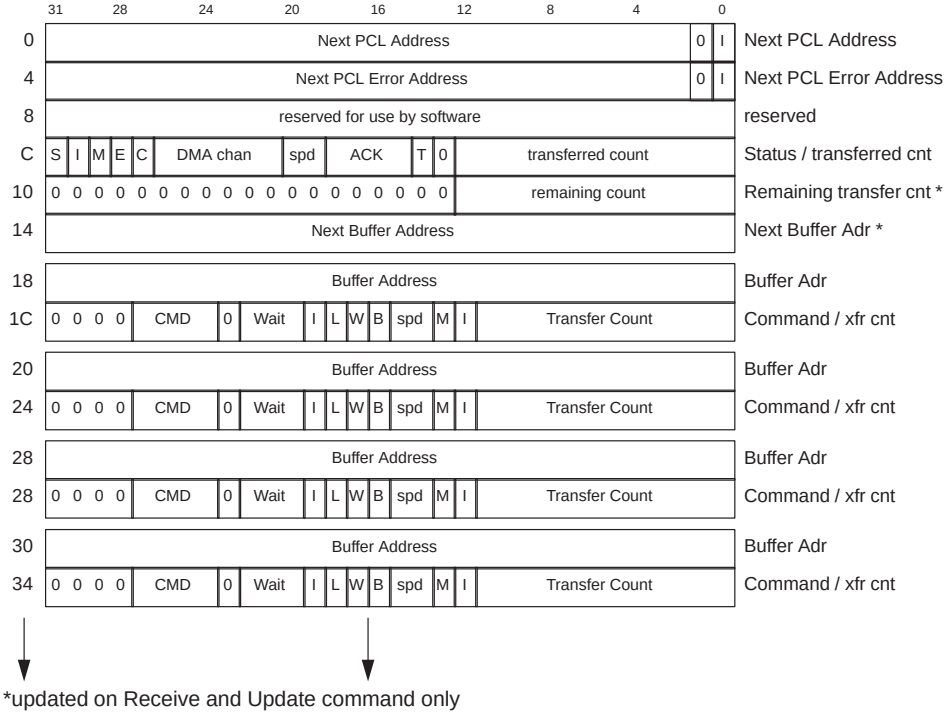


Figure 7. Typical Program Control List (PCL)

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. By using these commands with an address within the local bus address space, the PCI bus can be bypassed and transfers can occur directly between 1394 and the local bus. Two more commands deal with moving data between the PCI bus and the auxiliary bus. Only one channel of the DMA can be assigned to asynchronous transmits, due to complications arising from retries on the 1394 bus.

Transfer commands:

- 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. The state of the LSB of the

ready register will reflect the status of the last buff bit (bit 18) of the last PCL scatter entry processed. This way one can determine whether or not the last buffer of a multi-entry scatter table contains data.

- 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. The remaining PCL transfer count and next data buffer address for the current scatter table entry are returned to the PCL in offsets 0x10 and 0x14. The state of the LSB of the ready register will reflect the last buff bit (bit 18) of the last PCL scatter entry processed. This way one can determine whether or not the last buffer of a multi-entry scatter table contains data.
- 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. 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. The state of the LSB of the ready register will reflect the last buff bit (bit 18) of the last PCL scatter entry processed. This way one can determine whether or not the last buffer of a multi-entry scatter table contains data.

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

- NOP – No operation. The DMA reads the command but performs no operation.
- LOAD – @SOURCE => TEMP. Read quad data from a source address and store it in a DMA temp location. This permits 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. This permits 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. This permits 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 0s to an address. This allows 1 DMA channel to inform a waiting DMA channel to continue execution.
- STORE1 – FFFFFFFF ≥ @DESTINATION. Write all 1s to an address. This allows 1 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.
- SWAP & COMPARE – Swap the 16-bit halves of the temp register then compare the contents to a 16-bit immediate value with a 16-bit mask and store the equality result in the ready register. (PCILynx Rev A and higher only)

- ADD – Add the current contents of the temp register to a 3-bit immediate value and store the results back into the temp register.

The application software programs the operation of a DMA channel by using a packet control list (PCL) data structure. This structure can reside in host PCI memory or in any memory on the local bus. PCLs which reside in memory on the local bus have the potential of executing faster because the PCI bus latency is avoided. Local bus accesses can be slower however since the maximum data width is 16 bits. Application software is responsible for constructing the PCLs and allocating memory for their storage. A PCL is organized as a contiguous set of memory locations that contains 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 is limited to 32 quadlets. As a minimum requirement, the PCL starting address is aligned to a quadlet boundary (2 address LSBs = 00). For optimal DMA performance, the PCL start address is recommended to be aligned on a cache line boundary. The data buffer pointers are, as a minimum requirement, 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 active DMA channel fetches the commands and control parameters from the PCL, and uses them to configure itself to perform the command or transfer. Table 1 defines the format of a PCL for the transfer commands. Table 2 defines the format of a PCL for auxiliary commands.

Table 1. 1394 TRANSFER Packet Control List Format

OFFSET	PCL CONTENTS		DMA CHANNEL ACCESS PERFORMED
0x0	Next PCL address		Read
0x4	Next PCL address after an asynchronous transmit error.		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 and LBUS 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 equal 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.

ASYNCHRONOUS TRANSMIT ERROR ADDRESS OFFSET 0X4 **Note: This PCL address points to an alternate PCL queue that an active DMA channel will branch to if an error occurs that sets the Ack_Type bit in the status word during an asynchronous transmit command. The intent is to allow software to take some alternate action in the event of an error which may require subsequent commands to this device to be skipped. These include retry overruns, FIFO underruns, CRC error on the returned ack (not the payload), internal FIFO errors, and a corrupted header.**		
BIT NO.	**BIT NAME**	**DESCRIPTION**
31–04		Address of next PCL
03–02		These bits should equal 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.

STATUS AND TRANSFERRED COUNT OFFSET 0XC			
BIT NO.	**BIT NAME**	**DESCRIPTION**	
31	Self ID	Set when a self ID packet has been received by this channel. Refer to the definition of the receive comparator registers for how a channel is enabled for self ID reception.	
30	ISO MODE	The received packet was an isochronous packet.	
29	Mst Err	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 bits are cleared in the DMA command register (see register definitions) and an DMA_HLT interrupt (see interrupt status register) will be generated if enabled.	
28	Pkt Err	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.	
27	Pkt Cmp	Packet complete. Written by the DMA upon completion of this packet.	
26–21	Receive Dma_Cha[5–0]	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 return 0s for other commands.	
		Receive Dma_Cha[5–0]	DMA Channel Number
		0 0 0 0 0 0	0
		0 0 0 0 0 1	1
		0 0 0 0 1 0	2
		0 0 0 0 1 1	3
		0 0 0 1 0 0 Others	4 reserved

STATUS AND TRANSFERRED COUNT OFFSET 0XC (CONTINUED)		
BIT NO.	**BIT NAME**	**DESCRIPTION**
20–19	Rcv_Speed[1–0]	The speed at which the packet was received for asynchronous or isochronous transfers. Valid only for channels programmed for receive operations. These bits return 0s for other commands. 00 = 100 Mbps 01 = 200 Mbps 10 = 400 Mbps
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 0s 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 bits 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 an ACK_TIMEOUT 0010 = Link reported a FIFO underrun 0011 = Link reported a CRC error on a received 1394 ack packet 0100 = DMA received an end of packet token while expecting a start of packet token. Catastrophic internal error. 0101 = No expected end-of-receive packet 0110 = Pipelined asynchronous transmit command encountered a command other than another asynchronous 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. Refer to Ack definitions above.
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 (header + payload) for this packet. These bits are indeterminate for asynchronous transmits due to the potentially pipelined nature of asynchronous XMT commands. The count will also include any retried packets during asynchronous receives. These bits are written with 0s after completion of auxiliary commands.

REMAINING TRANSFER COUNT OFFSET 0x10		
BIT NO.	**BIT NAME**	**DESCRIPTION**
31–16	0	Written with 0s by the DMA for the RCV_AND_UPDATE command
15–13	Offset	For a RCV_AND_UPDATE, LBUS _TO_PCI, and PCI_TO_LBUS commands, 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.
12–00	Remaining count	For a RCV_AND_UPDATE, LBUS _TO_PCI, and PCI_TO_LBUS commands, 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, LBUS _TO_PCI, and PCI_TO_LBUS commands, 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.

<table>
<tr><th colspan="8">TRANSFER COMMAND DATA BUFFER0 CONTROL, BYTE COUNT</th></tr>
<tr><th>BIT NO.</th><th>BIT NAME</th><th colspan="6">DESCRIPTION</th></tr>
<tr><td>31–28</td><td>reserved</td><td colspan="6">These bits are set to all 0s.</td></tr>
<tr><td rowspan="9">27–24</td><td rowspan="9">CMD3–0</td><td>CMD3</td><td>CMD2</td><td>CMD1</td><td>CMD0</td><td colspan="2">Command</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td colspan="2">RCV. (1394 FIFO to memory)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td colspan="2">RCV_AND_UPDATE</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td colspan="2">XMT. (Memory to 1394 FIFO)</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td colspan="2">UNFORMATTED XMT</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td colspan="2">PCI_TO_LBUS</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td colspan="2">LBUS_TO_PCI</td></tr>
<tr><td colspan="6">A packet control list queue must have consistent RCV or XMT commands. i.e., the transfer direction must be consistent.</td></tr>
<tr><td>23</td><td>reserved</td><td colspan="6">This bit is set to 0.</td></tr>
<tr><td rowspan="11">22–20</td><td rowspan="11">Wait Sel 2–0</td><td colspan="6">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>Wait Sel 2</td><td>Wait Sel 1</td><td>Wait Sel 0</td><td colspan="3">Wait condition</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td colspan="3">No wait. Continue execution.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td colspan="3">Wait for DMA ready register = 1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td colspan="3">Wait for DMA ready register = 0</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td colspan="3">Wait for external ready pin RDY = 1</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td colspan="3">Wait for external ready pin RDY = 0</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td colspan="3">Wait for GPIO port 2 to go active</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td colspan="3">Wait for GPIO port 3 to go active</td></tr>
<tr><td colspan="3">Others</td><td colspan="3">Reserved, do not use.</td></tr>
<tr><td>19</td><td>Int</td><td colspan="6">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. This bit should be set in the first transfer command data buffer control word of any PCL with multiple scatter table entries if interrupts are to be generated. The interrupt bit location for subsequent scatter table entries are a don’t care.</td></tr>
<tr><td>18</td><td>Last Buff</td><td colspan="6">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="6">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="6">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="6">1394 transmit speed code. Specifies the transmission speed of an asynchronous or isochronous transmit packet.
xmt_spd_code[1–0] = 00–100 Mbps
xmt_spd_code[1–0] = 01–200 Mbps
xmt_spd_code[1–0] = 10–400 Mbps
The value of this field is only valid for DMA transmit commands and are ignored for commands other than XMT or UNFORMATTED XMT.</td></tr>
</table>

TRANSFER COMMAND DATA BUFFER0 CONTROL, BYTE COUNT (CONTINUED)		
BIT NO.	**BIT NAME**	**DESCRIPTION**
13	Multi ISO packet per cycle start	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. This bit is ignored for commands other than XMT or UNFORMATTED XMT.
12	Transmit ISO mode	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. This bit is ignored for commands other than XMT or UNFORMATTED XMT.
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. This count along with the last buff bit (bit 18) is used to determine the size of a transmitted packet. For receives, the sum of the scatter entry counts should be equal to or greater than the entire packet size.

TRANSFER COMMAND DATA BUFFER (0 TO N) ADDRESS POINTER		
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		
BIT NO.	**BIT NAME**	**DESCRIPTION**
31–19	Reserved	These bits are set to all 0s.
18	Last buff	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	This bit is set to 0.
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 are set to all 0s.
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. This count along with the last buff bit (bit 18) is used to determine the size of a transmitted packet. For receives, the sum of the scatter entry counts should be equal to or greater than the entire packet size.

Table 2. 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

OFFSET	PCL CONTENTS	DMA CHANNEL ACCESS PERFORMED
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 are set to all 0s.				
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	SWAP & COMPARE TEMP[31–16] => TEMP[15–0] TEMP[15–0] => TEMP[31–16] (TEMP ^ BUFFER => READY) (PCILynx Rev A and higher only)
		@DESTINATION, and @SOURCE addresses are contained in the next word. TEMP is the DMA previous address register.				
23	Reserved	This bit is set to 0.				

AUXILIARY COMMAND OFFSET 0x18, 0x20... etc. (CONTINUED)

BIT NO.	BIT NAME	DESCRIPTION
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.

BIT NO.	BIT NAME	DESCRIPTION
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 valid for any auxiliary command.
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 are set to all 0s.

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 [0]

BIT NO.	BIT NAME	DESCRIPTION
31–00	Don't care	Read but not used by the DMA
	LOAD [3]	
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 [4]	
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 [B]	
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 [5]		
BIT NO.	**BIT NAME**	**DESCRIPTION**
31–00	@DESTINATION	Address where data of 00000000 will be written.
STORE1 [6]		
31–00	@DESTINATION	Address where data of FFFFFFFF will be written.
COMPARE [E]		
SWAP & COMPARE [F]†		
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 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 [D]		
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 [7]		
31–00	DESTINATION	This address is loaded into the CURRENT PCL ADDRESS if the conditions are met as specified in the condition code field.

† PCILynx Rev A and higher only

Application software programs a DMA channel to transfer multiple 1394 data packets by chaining together multiple PCLs into a packet control list queue. A queue is 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.

It is possible to combine auxiliary commands and transfer commands in the same PCL with the following restrictions:

- All auxiliary commands must precede any transfer commands and will be executed in sequence.
- If a 1394 busy ack status is sent by the asynchronous receiver or detected by the asynchronous transmitter, then the ENTIRE PCL including all auxiliary commands within the PCL will be re-executed.
- If any asynchronous transmit command is followed by a PCL with any command other than another asynchronous transmit, then the wait-for-status-before-continue bit in the command word must be set.
- The status word and the DMA interrupt will reflect the last command that was executed by the DMA in that PCL. For example, if a branch command caused the DMA to exit the PCL, then the status and interrupt would be based on the branch command.

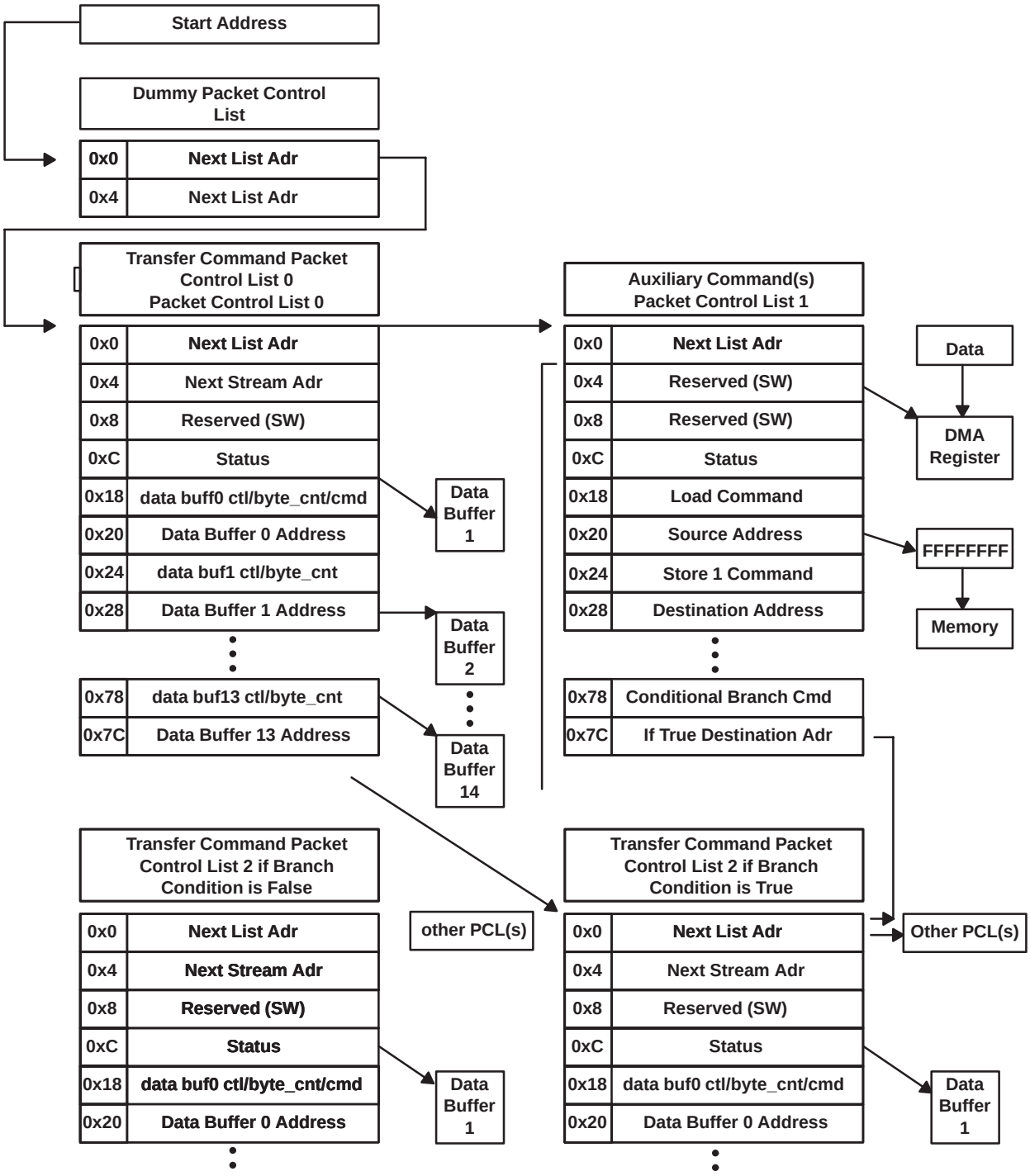


Figure 8. Example PCL Queue

4.2.3.1 DMA Engine

The DMA engine function implements the state machine logic for fetching the control parameters and data buffer pointers from the PCL. The state machine logic or packet processor uses these parameters to control the transfer of data to or from the data buffers. Table 3 defines the DMA channel priority assignment. Figure 9 defines the overall flow of each DMA channel.

Table 3. 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

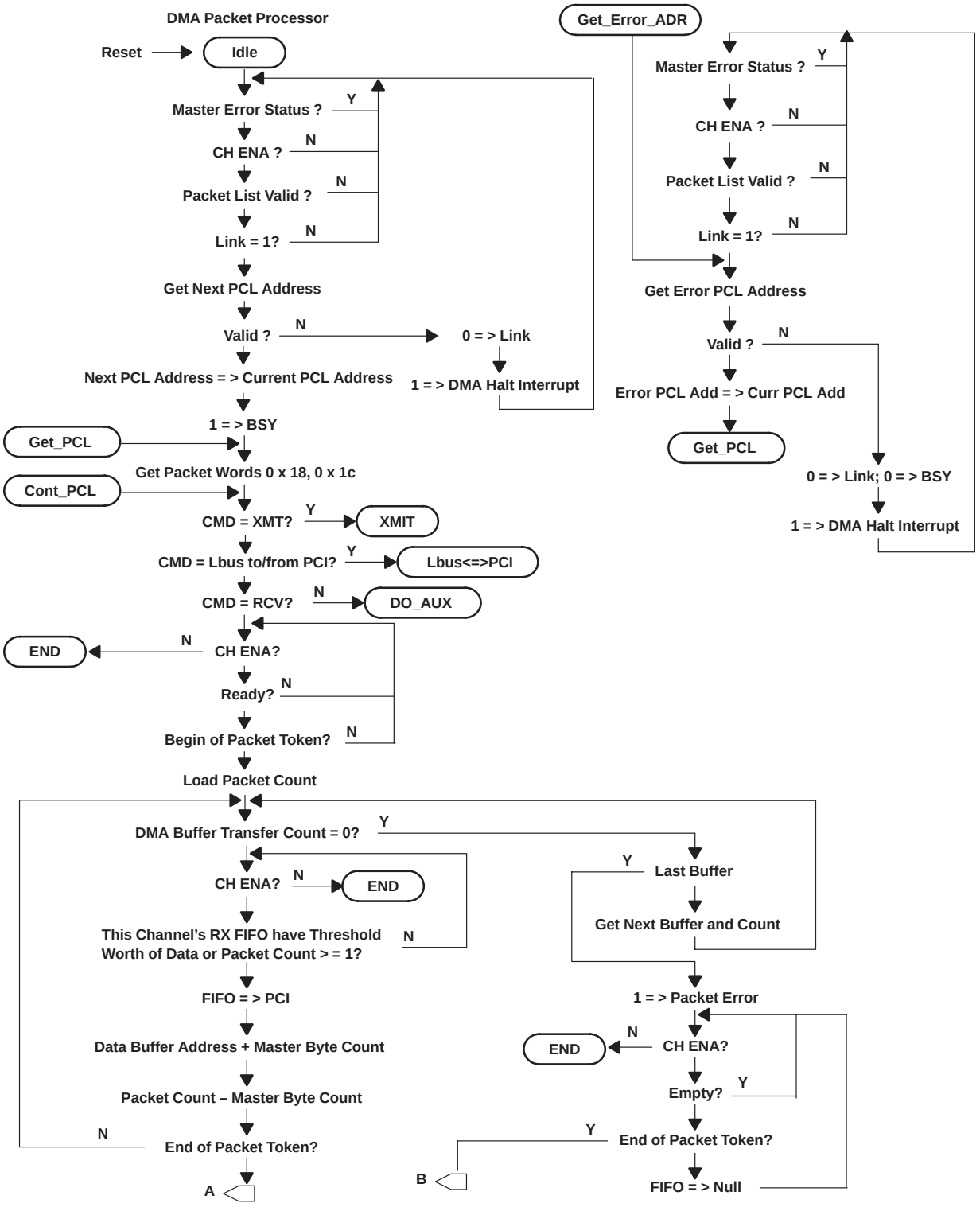


Figure 9. State Machine Flowchart

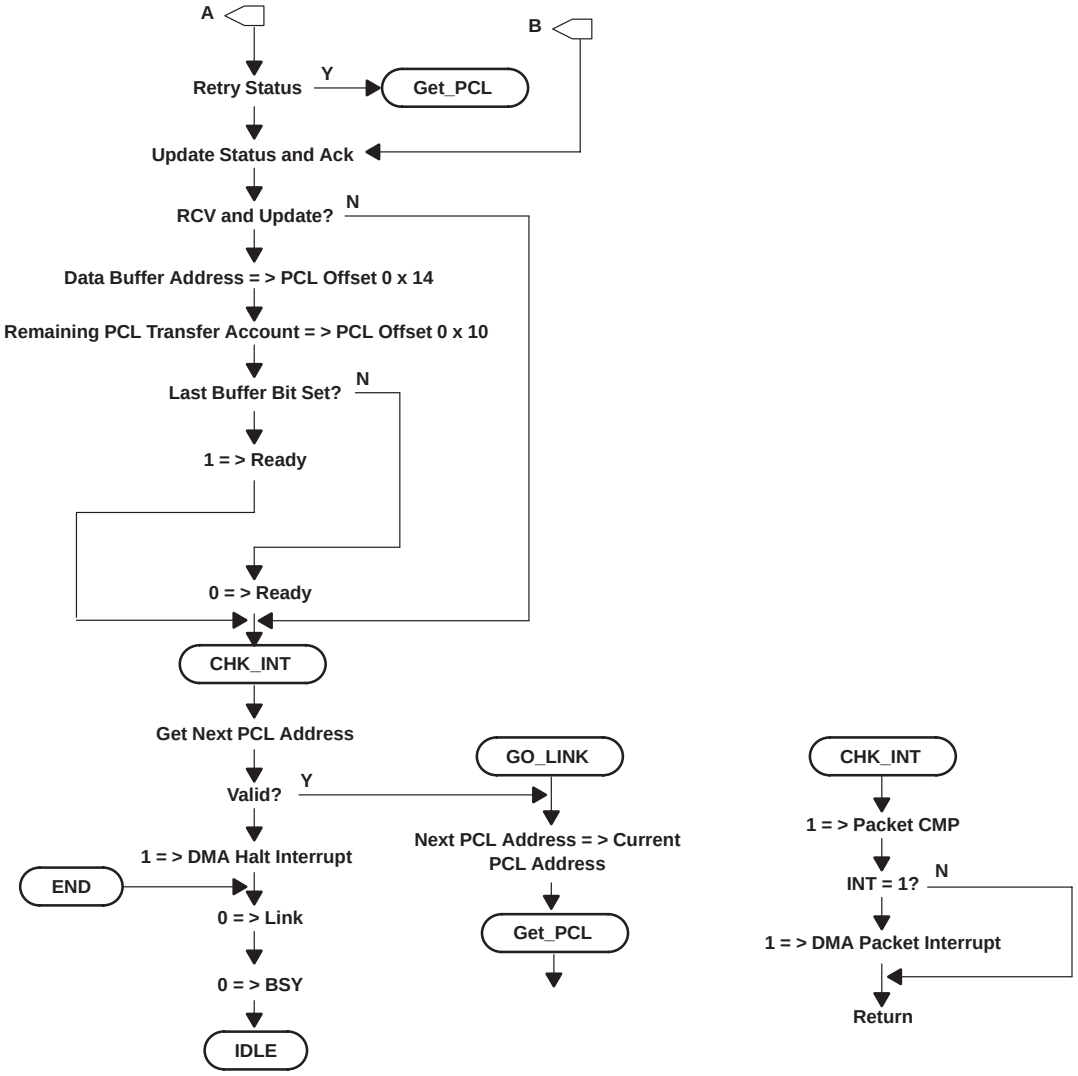


Figure 9. State Machine Flowchart (Continued)

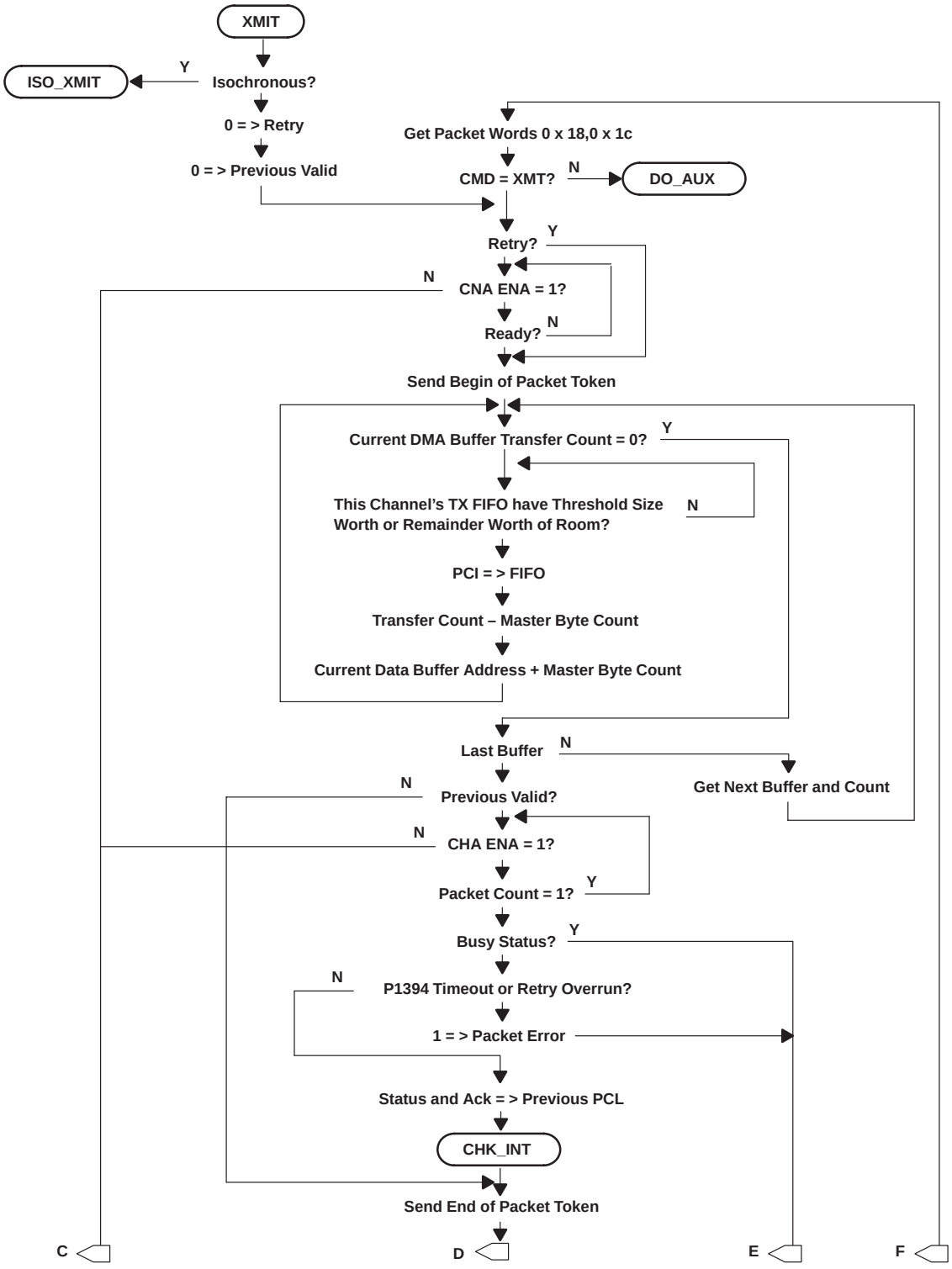


Figure 9. State Machine Flowchart (Continued)

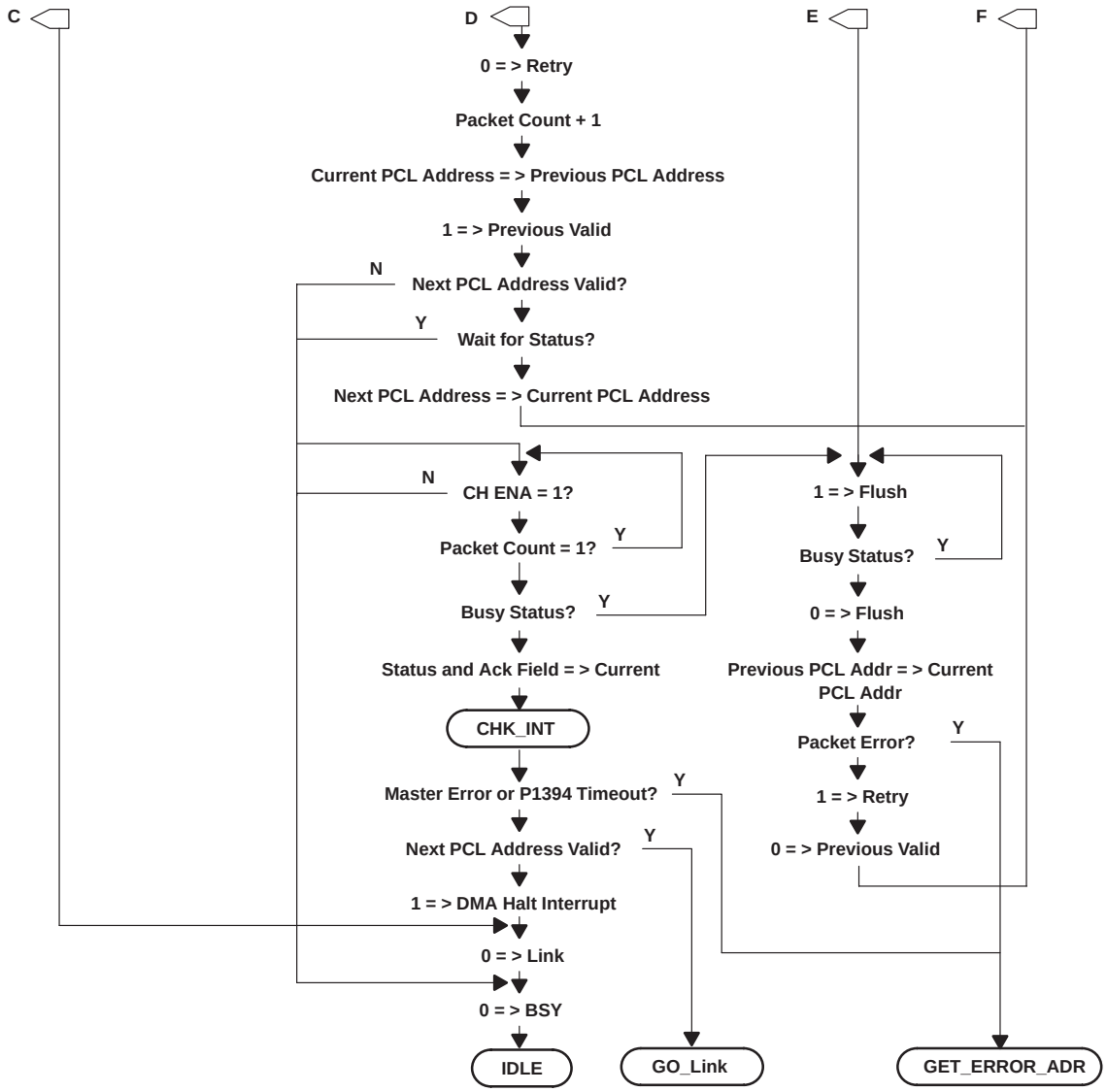


Figure 9. State Machine Flowchart (Continued)

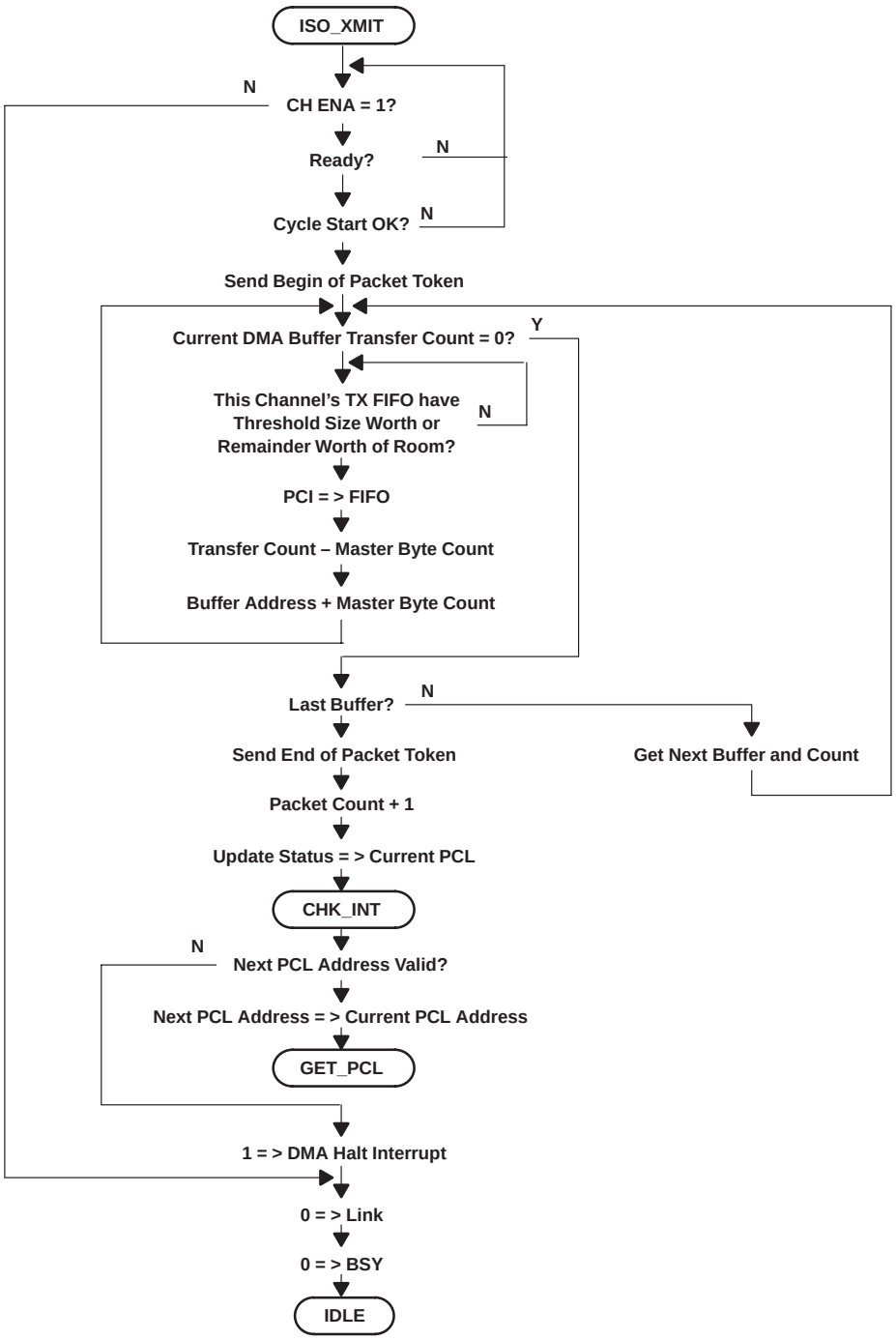


Figure 9. State Machine Flowchart (Continued)

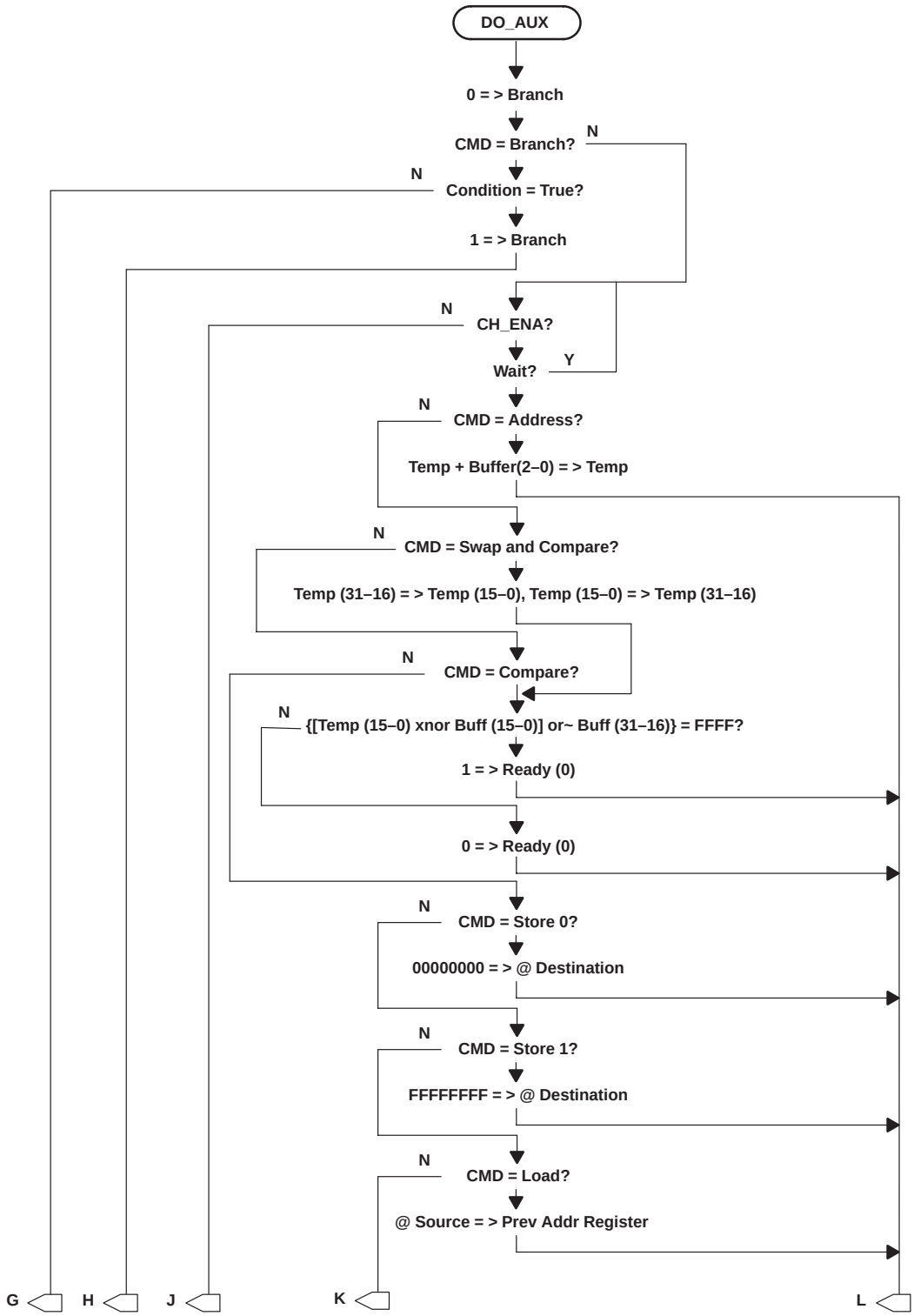


Figure 9. State Machine Flowchart (Continued)

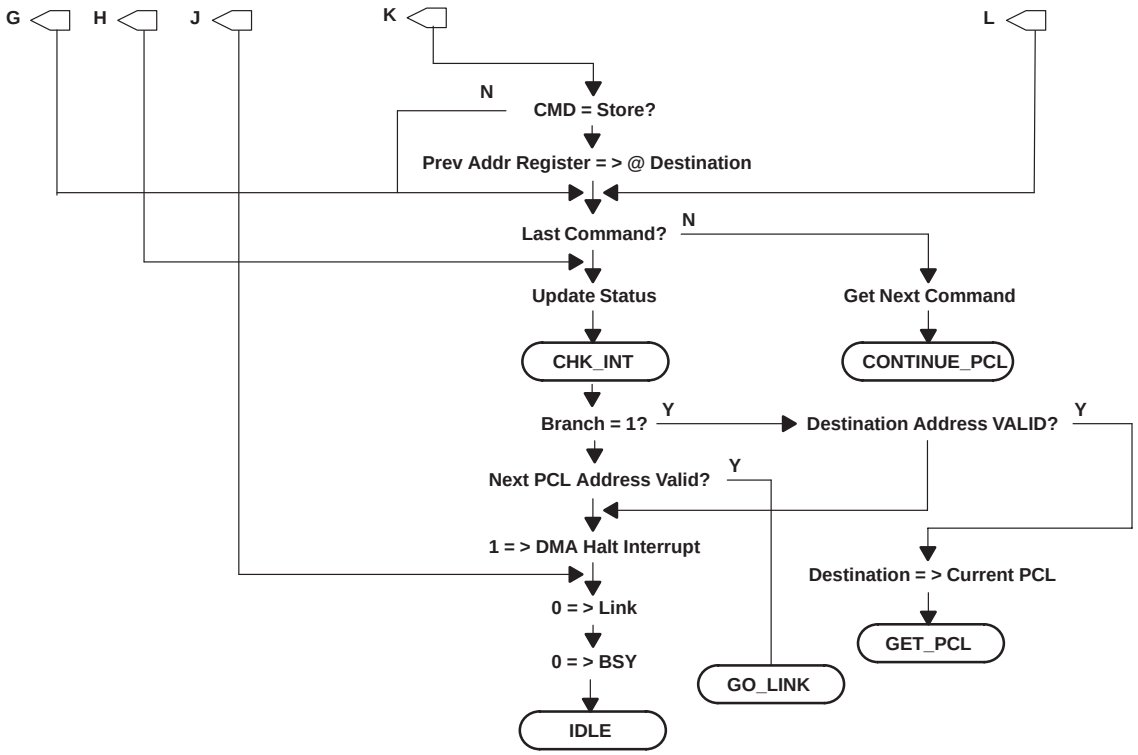
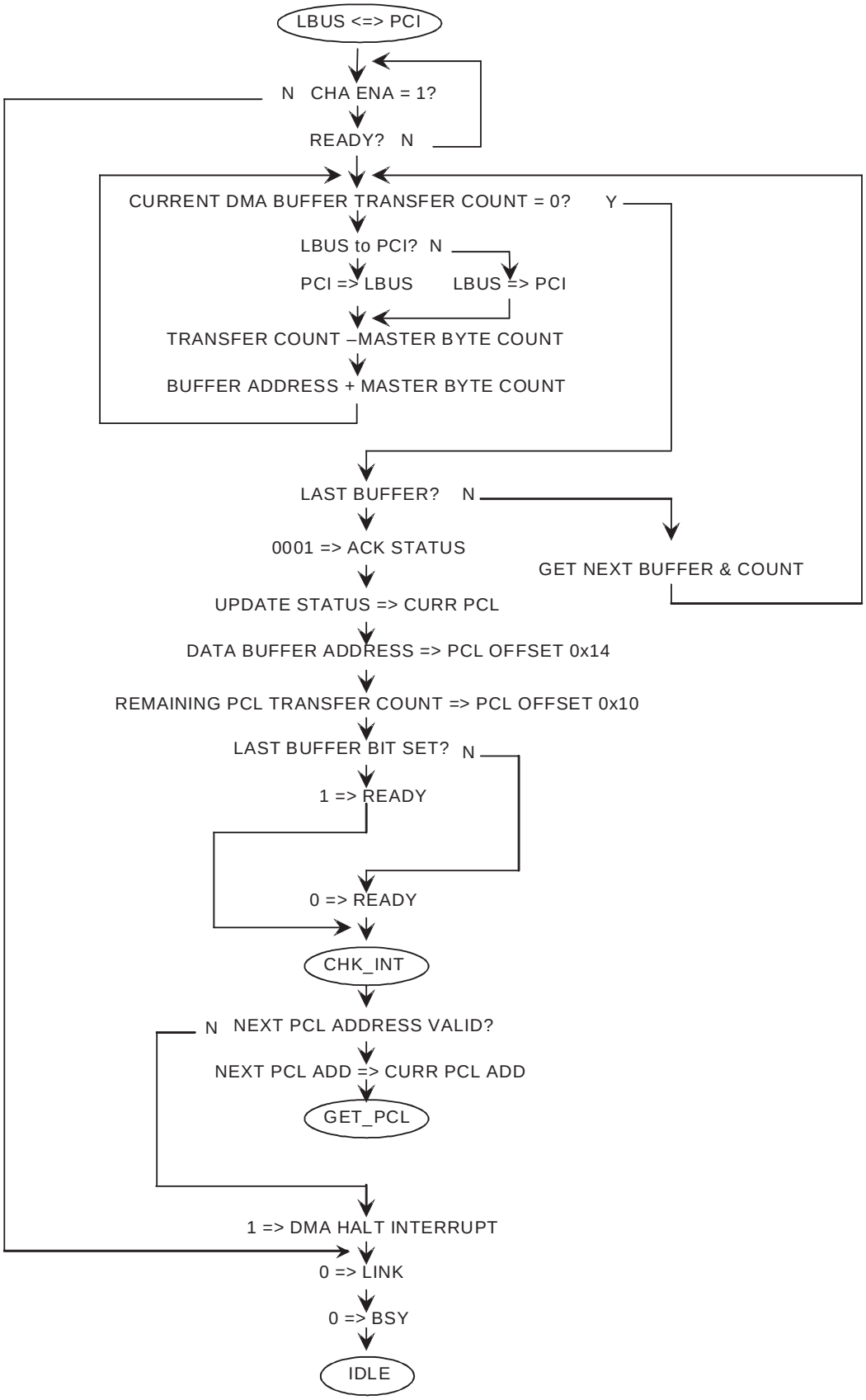


Figure 9. State Machine Flowchart (Continued)


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
DATA BUFFER ADDRESS => PCL OFFSET 0x14
REMAINING PCL TRANSFER COUNT => PCL OFFSET 0x10
LAST BUFFER BIT SET? N
1 => READY
0 => READY
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 five independent DMA channels all running concurrently. The actual implementation utilizes one main control state machine which multiplexes between the five 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 current packet control list 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 current packet control list 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 current packet control list address register, get the next address and, if valid, will make this the current PCL address and begin execution. If this address is invalid, then the link bit is cleared in the DMA control register, a DMA halted interrupt is generated, if enabled, 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 0x1C. A check is then made to determine whether the command is a receive, transmit, PCI to/from LBUS, or auxiliary command.

4.2.3.1.1 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.

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, then 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 the number of bytes in the receive FIFO reaches a high water mark. This high water mark is equal to the value specified in the lower bound field of the DMA global register. Refer to the register definition for further details.

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 remaining 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 for an asynchronous receive indicates a 1394 busy acknowledge, then the DMA reacquires 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 bit is set, and it is then written to memory in the PCL status word at PCL offset 0xC along with the total number of bytes (including retries) transferred for this PCL. If the INT bit is set in the data buffer control word, 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, if enabled, with associated status (DMA_HLT[x]) in the interrupt status register, and the channel becomes idle.

4.2.3.1.2 DMA Asynchronous Transmit Operation

Asynchronous transmits are determined after a valid PCL pointer has been written to the current 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 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 checks 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. 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, then 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 remaining data buffer transfer length bits in the DMA control register are updated to reflect the current state of the transfer.

The DMA knows that the end of a packet has been reached 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 buff bit (bit 18) of the ctl/byte_cnt PCL word. If the previous packet address is valid, then 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 buffer control word, 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 is 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, then the FIFO is flushed as aforementioned and the transmit is attempted again.

If there was a PCI master error, 1394 transfer error, 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 an error that sets the Ack_Type bit, it may be possible that the host software will want to take some action other than continuing the transmit. 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 asynchronous transmit error address entry of the PCL at offset 0x4 to point to the first PCL of the next stream of transmit data (next asynchronous transmit node) or to some other error processing PCL. If the asynchronous transmit error 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).

Once a status with Ack_Type is set for an asynchronous transmit, the DMA must be given a valid asynchronous transmit error address (offset 0x4) before it will continue. This means that even if a different address is loaded into the current PCL address register, offset 0x4 will still be used for fetching the next PCL. If one changes the current PCL address register to point to say, for example, a dummy PCL, then offset 0x4 of this dummy PCL must still contain a valid pointer.

If the DMA halts due to DMA_HLT[x], and the next PCL stream entry was invalid, then rewriting the next PCL stream entry is necessary since the DMA is in the GET_NEXT_STREAM state of Figure 9 and the DMA state machine is ignoring the next list adr. Always setting 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 asynchronous XMT channel that invokes the next PCL stream entry due to an error.

If there was no retry, timeout, or FIFO underrun, then the DMA will update the DMA status register with the 1394 acknowledge status passed from the link layer controller, the PKT CMP bit 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 buffer control word, 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, then the link and BSY bits are cleared in the DMA control register, a DMA halted interrupt is generated, if enabled, for this channel with associated status (DMA_HLT[x]) in the interrupt status register, and the channel becomes idle.

4.2.3.1.3 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 Figure 9. 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. Since only one packet per channel is allowed during a cycle start period on the bus, the DMA will also optionally wait for an 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. Figure 10 illustrates this concept. The waiting for the cycle start framer can be disabled by the multi-isochronous packet per cycle start bit in the PCL data buf0 ctl/byte_cnt/cmd word. The effect of setting the multi-isochronous bit is global and can affect other isochronous transmit channels so all isochronous channels should set the multi-isochronous bit to the same value to prevent otherwise unpredictable behavior.

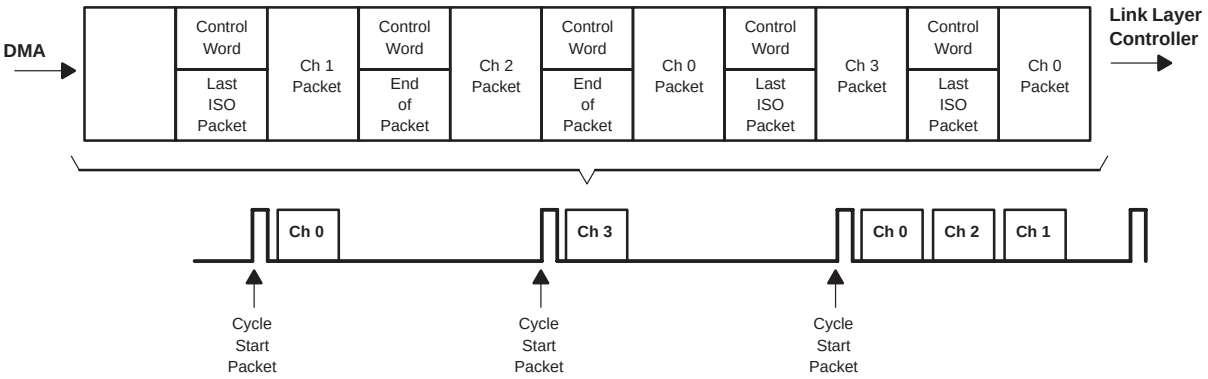


Figure 10. Isochronous Transmit Packet Framing

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, then 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 remaining 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 buff bit (bit 18) 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 is 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 bit 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 buffer control word, 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, then the link and BSY bits are cleared in the DMA control register, a DMA halted interrupt is generated, if enabled, for this channel with associated status (DMA_HLT[x]) in the interrupt status register, and the channel becomes idle.

4.2.3.1.4 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 buff 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 proceeds 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 word. 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, then 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 remaining 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 buff bit (bit 18) of

the ctl/byte_cnt PCL word, then the DMA knows that the end of the transfer has been reached. The remaining transfer count and next buffer address are written to PCL offsets 0x10 and 0x14, respectively. The DMA will update the DMA status register with status of 0x0001, the PKT CMP bit 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 buffer control word, 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 makes 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, if enabled, for this channel with associated status (DMA_HLT[x]) in the interrupt status register, and the channel becomes idle.

4.2.3.2 DMA Registers

This function implements a control and status register set for controlling and monitoring the status of each DMA channel. The register set is 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 refetch 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 LSB 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 LSB 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 asynchronous 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.

4.2.3.3 DMA Channel Global Issues

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

- If a DMA channel programmed for asynchronous receives errors with a packet error or master error, then the asynchronous transmit error address at PCL offset 0x4 must point to a valid PCL before that channel's execution can continue.
- Only one DMA channel may be programmed for asynchronous transmits due to complications with 1394 retries.
- If any isochronous transmit channel has the multi-isochronous bit set in the command word of any of its PCL's, then all other isochronous 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
  - PCL's using these commands must reload the AUX address register as necessary.

4.2.4 FIFO Logic

This function is designed around a single 1024 X 33 bits clocked dual-port RAM. The RAM is partitioned into 3 logical FIFOs. Each FIFO is programmable in size from 0 to 255 decimals. Each decimal value represents 4 quadlets (16 bytes). For a given combination of FIFO sizes, the sum total of the 3 FIFO sizes is less than or equal to 1024 quadlets. Each FIFO is assigned to a 1394 transfer mode as shown in Table 4. Figure 11 provides a high level functional block diagram of the FIFO.

Table 4. FIFO Assignments to a 1394 Transfer Mode

FIFO	1394 RECEIVER	1394 TRANSMITTER	ACTIVE DMA CHANNEL
General receive FIFO (GRF)	Writes asynchronous or isochronous packets received from the 1394 bus to the FIFO	NOP	Reads the asynchronous or isochronous packets from the FIFO and writes them to host memory
Asynchronous transmit FIFO (ATF)	NOP	Reads asynchronous packets from the FIFO and transmits them over the 1394 bus	Reads the asynchronous packets from host memory and writes them into the FIFO
Isochronous transmit FIFO (ITF)	NOP	Reads isochronous packets from the FIFO and transmits them over the 1394 bus	Reads the isochronous packets from host memory and writes them into the FIFO

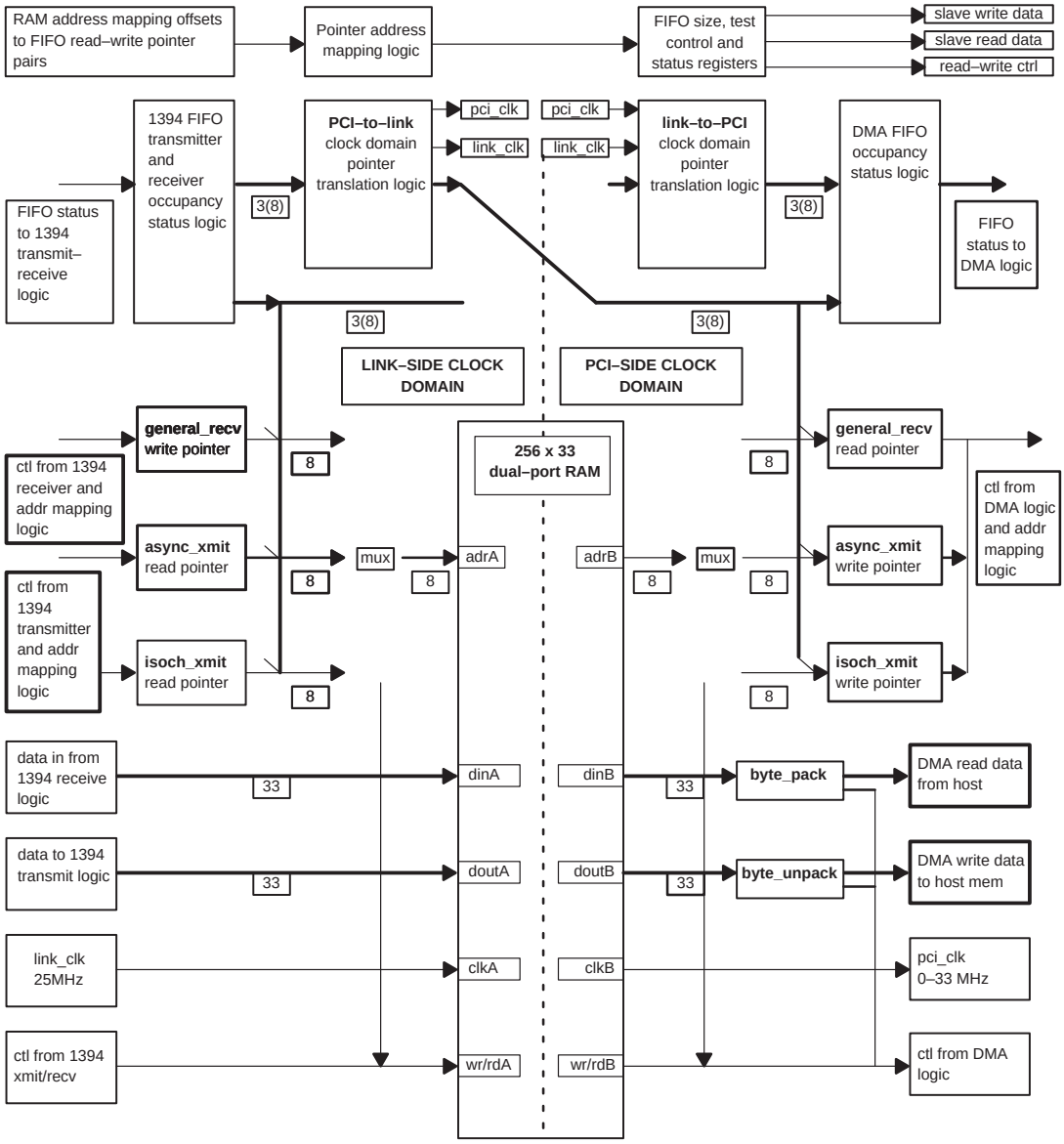


Figure 11. FIFO High Level Functional Block Diagram

4.2.4.1 General Receive FIFO

The general receive FIFO (GRF) is comprised of the read and write pointer pair as shown in Figure 11. These pointers are used in accessing the dual-port RAM. Each pointer counts in the range from 0 to its FIFO_size_value minus 1. The RAM addressing range for each pointer is set by logic which generates an offset value. The offset is added to the value of the pointer to map it to a unique range of RAM addresses. The read pointer is 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 is 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 are connected to their respective sides of the RAM through a multiplexer network.

4.2.4.2 Asynchronous Transmit FIFO

The asynchronous transmit FIFO (ATF) is comprised of the read and write pointer pair as shown in Figure 11. These pointers are used in accessing the dual-port RAM. Each pointer counts in the range from 0 to its FIFO_size_value minus 1. The RAM addressing range for each pointer is set by logic which generates an offset value. The offset is added to the value of the pointer to map it to a unique range of addresses. The write pointer is 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 is 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 are connected to their respective sides of the RAM through a multiplexer network.

4.2.4.3 Isochronous Transmit FIFO

The isochronous transmit FIFO (ITF) is comprised of the read and write pointer pair as shown in Figure 11. These pointers are used in accessing the dual-port RAM. Each pointer counts in the range from 0 to its FIFO_size_value minus 1. The RAM addressing range for each pointer is set by logic which generates an offset value. The offset is added to the value of the pointer to map it to a unique range of addresses. The write pointer is used by the active DMA channel to write isochronous packets—that it reads from host memory—into the PC-side of the RAM. The read pointer is 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 are connected to their respective sides of the RAM through a multiplexer network.

4.2.4.4 FIFO Status Logic

This function implements 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 samples the current value of each pointer on the link side of the FIFO and translates these samples from the link clock domain over to the PCI clock domain. Each translated link-side pointer is compared to its corresponding PCI-side pointer to generate an occupancy status for each FIFO. This status is 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 samples the current value of each pointer on the PCI-side of the FIFO and translates these samples from the PCI clock domain over to the link clock domain. Each translated PCI-side pointer is compared to its corresponding link-side pointer to compute an occupancy status for each FIFO. This status is used by the 1394 transmit-receive logic to pace the transfer of data between the1394 bus and the FIFO.

4.2.4.5 Pointer Dual-Port Address Mapping Logic

This function uses 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 is generated in accordance with the equations as shown below.

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)

4.2.4.6 Byte Pack Logic

This function implements 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 consists of four 8-bit wide registers and four 8-to-1 multiplexers. Each register-mux pair corresponds to a byte lane. The input of each register connects to an input byte lane which is switched by the active DMA channel to host memory. The output of each mux connects to an output byte lane, which drives the FIFO. For each 8-to-1 multiplexer, four inputs connect in a one-to-one correspondence to each register output. The remaining four inputs connect in a one-to-one correspondence to each register input. This configuration allows byte-aligned DMA read data from the four input byte lanes, to be cross-point switched in a different order to the four output byte lanes. The control of the byte lane multiplexers is performed by the active DMA read channel.

4.2.4.7 Byte Unpack Logic

This function implements 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 consists of four 8-bit wide registers and four 8-to-1 multiplexers. Each register-mux pair corresponds to a byte lane. The input of each register connects to an input byte lane, which is driven from the FIFO. The output of each multiplexer connects to an output byte lane which is switched by the DMA channel to the host memory. For each of the 8-to-1 multiplexers, four inputs connect in a one-to-one correspondence to each register output. The remaining four inputs connect in a one-to-one correspondence to each register input. This configuration allows 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 is performed by the active DMA write channel.

4.2.4.8 FIFO Control and Status Registers

This function implements the control and status register set of the FIFO logic. These registers are implemented as specified in section 5.4, *FIFO Control and Status Register Definitions*. 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 provides three size parameters for programming the size of the ITF, ATF, GRF. This register is accessed via a PCI-slave read or write operation.
- PCI-side FIFO pointer write-read port—Provides 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—Provides 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 causes the data quadlet to be pushed onto the top of the GRF. A 32-bit slave read from this port causes 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 causes the data quadlet to be pushed onto the top of the ATF. A 32-bit slave read from this port causes 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 causes the data quadlet to be pushed onto the top of the ITF. A 32-bit slave read from this port causes a data quadlet to be popped off the top of the ITF.
- FIFO control token status read port—A slave read from this port returns the value of bit 33 of the last data quadlet that was popped from one of the three FIFOs that was previously accessed.
- FIFO diagnostic test and control register—Provides a PCI-slave read/write port for software to configure the FIFO logic for diagnostic testing and control its operation.
- Transmit FIFO threshold register—Provides a PCI-slave read/write port for software to set the transmit threshold for the asynchronous and isochronous transmit FIFOs.

4.2.5 1394 Link Layer Logic

This function implements the 1394 link layer control logic (LLC) as specified in section 6.0 of the IEEE 1394–1995 standard. This function controls the transmission and reception of 1394 packet data between the PCILynx FIFO and other devices on the 1394 bus. Figure 12 provides a high level functional block diagram of the link layer controller logic.

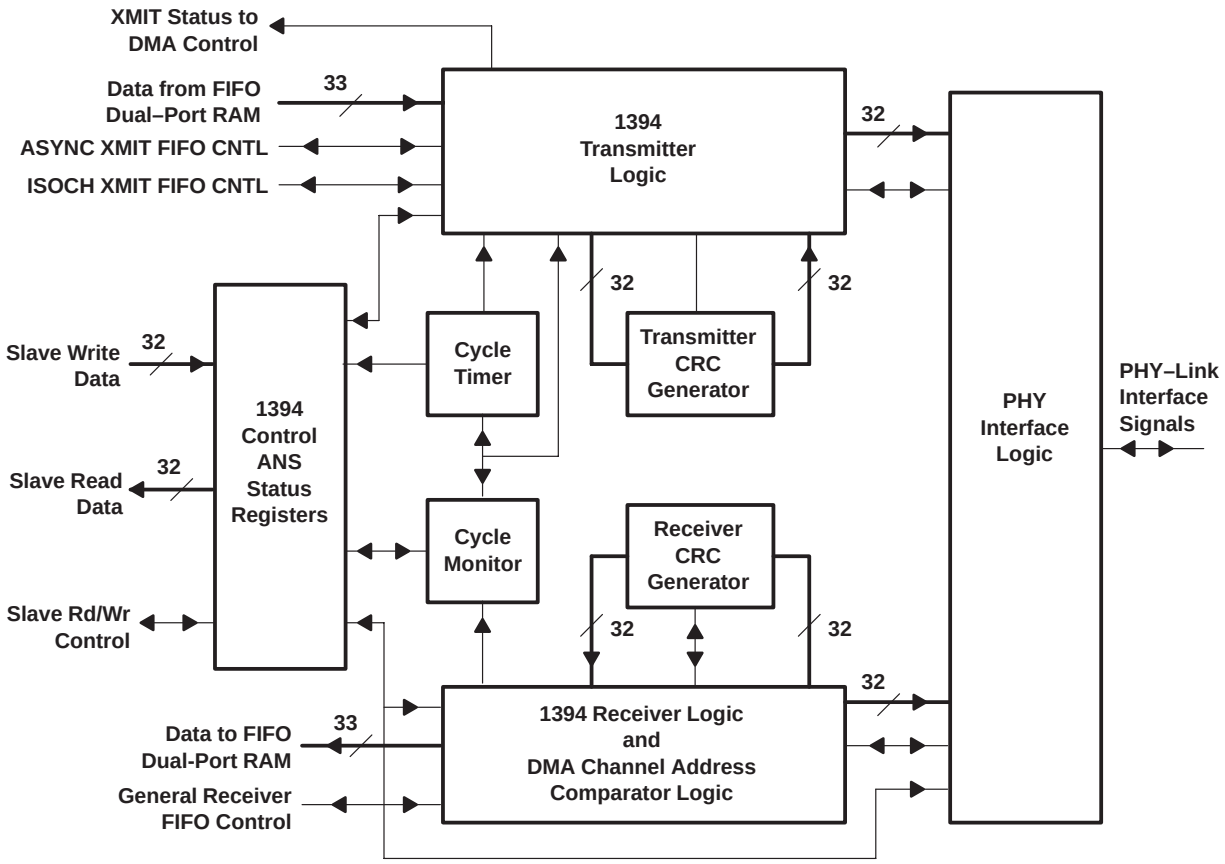


Figure 12. High-Level 1394 Link Layer Controller Block Diagram

4.2.5.1 1394 Link Layer Control and Status Registers

This function implements the control and status register logic required by application software to control the operation of the LLC and monitor its operation. This register set is implemented as specified in section 5.5, *1394 Link Layer Control and Status Register Definitions*. The following register set is implemented.

- 1394 bus number – node number register: Provides the interface for application software to program the bus and node numbers.
- 1394 link layer control register: Provides the interface for application software to control the operating mode of the LLC.
- 1394 link layer interrupt status register: Provides the interface for application software to decode the cause of interrupts generated by the LLC and provide a mechanism for clearing the interrupt status.
- 1394 link layer interrupt enable register: Provides 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: Provides 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 is used to time the transmission of cycle start packets every 125 μs.

- 1394 physical layer access register: Provides 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 provides the interface for application software to perform diagnostic testing of the 1394 LLC logic.
- DMA channel 4–0 word 0 receive packet compare value registers: There are five of these registers. Each register is assigned to a DMA channel comparator logic function. The DMA channel comparator matches 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 are specified by the mask value contained in the word 0 receive packet compare mask register.
- DMA channel 4–0 word 0 receive packet compare mask register: There are five of these registers. Each register is assigned to corresponding DMA channel comparator. The DMA channel compare logic uses 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 4–0 word 1 receive packet compare value registers: There are five of these registers. Each register is assigned to a DMA channel comparator logic function. The DMA channel comparator matches 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 are specified by the mask value contained in the word 1 receive packet compare mask register.
- DMA channel 4–0 word 1 receive packet compare mask register: There are five of these registers. Each register is assigned to corresponding DMA channel comparator. The DMA channel compare logic uses 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 specify the number of times the 1394 transmitter should retry the transmission of an asynchronous packet when a busy acknowledge is received from the destination node. This register is read/write by application software via PCI slave access.
- Busy retry transmit time interval register: The contents of this register specify the time interval that the transmitter must delay between successive retry attempts, when a busy ack is received for each attempt. This register is read/write by application software via PCI slave access.
- State machine vector register: The register provides software with the capability to monitor the state vector of each state machine implemented in the LLC.
- FIFO error counters: These counters count the underruns that occur on the asynchronous and isochronous transmit FIFOs during packet transmissions and the overruns occurring on the GRF during packet reception.

4.2.5.2 1394 Packet Transmit Control Logic

This function implements 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 conforms to the detail functional requirements as specified in section 6.3 of the IEEE 1394–1995 standard. The transmit control logic is designed to format the transmit packet formats listed in Appendix C and the FIFO control token formats listed in Appendix D. The following is high-level summary of the functions.

- 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 the IEEE 1394–1995 standard.
- 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 the IEEE 1394–1995 standard.
- 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.
- Input 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 retry transmissions using the single phase retryX protocol as specified in the IEEE 1394–1995 standard.
- Set the speed of packet transmission.

4.2.5.3 DMA Channel Receive Packet Comparator Logic

This function implements 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 implements five software programmable comparators. Each comparator is assigned to service a DMA channel. A comparator is comprised of a word 0 field select register, a word 1 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 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 two compare value registers 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 combines the select indications from the five comparators and generates a single comparator match indication to the 1394 receiver logic. The 1394 receiver logic uses 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 5.5, *1394 Link Layer Control and Status Register Definitions* for a detailed definition of the compare value and field select mask registers.

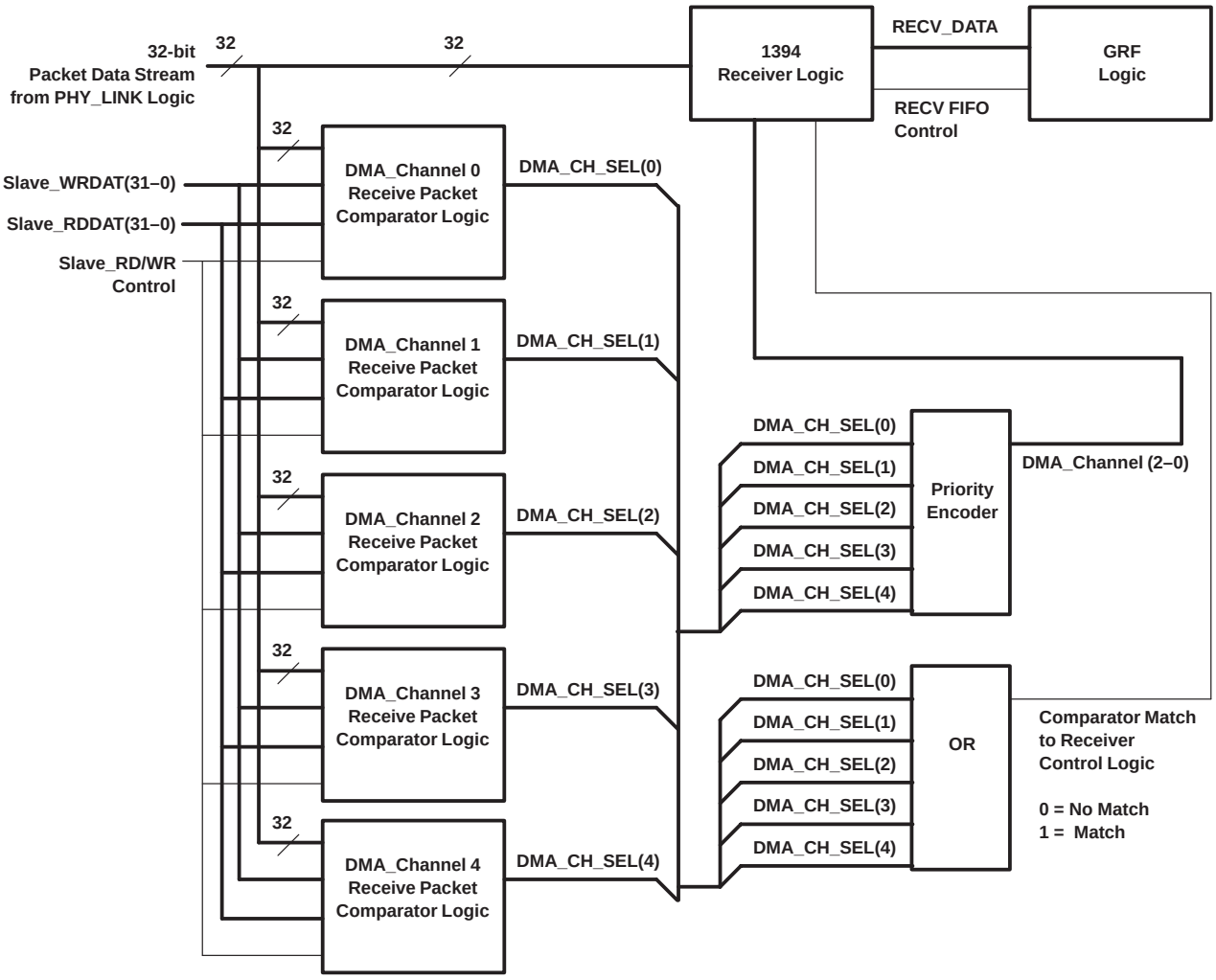


Figure 13. High-Level Functional Block Diagram of DMA Channel Receive Packet Comparator Logic

4.2.5.4 1394 CRC Logic

This function implements the logic for performing the following functions:

- Generate 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, generate 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.
- Generate 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.

- Generate 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.

4.2.5.5 1394 Packet Receiver Control Logic

This function implements the logic required to receive incoming 1394 packets. The design of the receiver control logic conforms to the detail functional requirements as specified in section 6.0 of the IEEE 1394–1995 standard. The following is high-level summary of the functions:

- 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, then the data CRC is checked.
- Load received packets into the general receive FIFO if the packet passes the addressing and CRC checks.
- Generate acknowledge on nonbroadcast asynchronous receive packets.
- Snoop 1394 bus traffic when snoop mode is enabled.
- Receive cycle start packets.
- Receive self-ID packets and load them into the general receive FIFO.

4.2.5.5.1 Snoop Mode

When the SNOOP_ENABLE bit is set in the link layer control register, the PCILynx enters a special mode where only DMA channel 0 is used for all received packets. The RCV_COMP_VALID bit is ignored as well as various comparator bits. All packets seen on the 1394 bus are received into DMA channel 0 PCLs.

The SNOOPED packet quadlets contain all data quadlets observed on the 1394 bus, including the 1394 header quadlets, header CRC quadlet, any payload quadlets, and the payload CRC quadlet. These CRC quadlets are not normally received into the FIFO. The header CRC follows the 1 quadlet isochronous header or the 3 or 4 quadlet asynchronous header. The payload CRC follows the last payload quadlet, and is aligned on a quadlet boundary. Runt packets, PHY configuration, SELFID, and Link-on, do not have any additional CRCs that will show up in the received packet.

A SNOOPED ACK quadlet containing the ACK information as seen on the 1394 bus is inserted into the FIFO following the received packet. If a SNOOPing node is specifically addressed by an asynchronous packet, then the SNOOPing node will never return an ACK on the 1394 bus and will insert a SNOOPED ACK of 0 into the receive FIFO. A SNOOPED ACK of 0 is always stored for isochronous packets.

4.2.5.6 Cycle Timer Logic

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

4.2.5.7 Cycle Monitor Logic

This function implements the logic for performing the cycle monitor function. The cycle monitor is used to support isochronous data transfers. It monitors the LLC activity and handles the scheduling of isochronous activity.

When a cycle start packet is received or transmitted, the cycle monitor indicates the occurrence of these events by generating a cycle started or cycle received interrupt. The cycle monitor also detects missing cycle start packets and generates a cycle lost interrupt. When an isochronous cycle is completed, the cycle monitor asserts a cycle done interrupt. The cycle monitor signals the transmitter to send a cycle start packet when the CYCLEMASTER enable bit is asserted in the 1394 LLC control register.

4.2.5.8 PHY-Link Interface Logic

This function implements the logic for interfacing the PCILynx to the physical layer chip. The design of this logic conforms to the requirements of the LINK-PHY interface specification in annex J of the IEEE 1394–1995 standard. This function provides the PCILynx with access to the physical layer services. The following high level functions are 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 100 Mbps, then the parallel data stream is converted into 2 serial data streams each running at 50 Mbps. For 200 Mbps, the parallel data stream is converted into 4 streams; at 400 Mbps, the parallel data stream is converted into 8 streams; each running at 50 Mbps.
- 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 generates 2 serial data streams to the PCILynx if it is receiving the packet at 100 Mbps, 4 streams at 200 Mbps, or 8 streams at 400 Mbps. The serial data streams are each clocked at 50 Mbps.
- Detect and receive serial status responses from the PHY and convert them into a parallel format. The status responses 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 PCILynx devices.

4.2.5.9 PHY-Link Interface Electrical Isolation

The PCILynx implements the electrical isolation logic for the electrical isolation mechanism specified in the IEEE 1394–1995 standard; however, the input receivers for the PHY signals do not meet the hysteresis requirements. Therefore, the $\overline{\text{link\_iso}}$ pin must be held at a high level.

5 Hardware Register Definitions

5.1 Memory and Configuration Address Space Register Map

OFFSET	PCI MEMORY ADDRESS SPACE 0	PCI CONFIGURATION ADDRESS SPACE
00	PCI Configuration, Miscellaneous, and Local Bus Registers	PCI Configuration, Miscellaneous, and Local Bus Registers
FC		
100	DMA Control and Status Registers	
9FC		
A00	FIFO Control and Status Registers	
AFC		
B00	1394 Link Layer and Physical Layer Status and Control Registers	
FCC		

	PCI Memory Address Space 1
00	PCILynx Local Bus RAM
FFFC	

	PCI Memory Address Space 2
00	PCILynx Local Bus AUX
FFFC	

	PCI Expansion ROM
00	PCILynx Local Bus ROM
FFFC	

Figure 14. Memory and Configuration Address Space Map

Table 5. PCI Address Offset Assignments for PCILynx 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			
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	PCI expansion ROM base address			
034	Zero			
038	Zero			
03C	Max_Latency	Min_Grant	Int_Pin	Int_Line
040	Miscellaneous control			
044	Serial EEPROM control			
048	PCI interrupt status			
04C	PCI interrupt enable			
050	PCI test			
054	Zero			
058	Subsystem access			

0B0	Local bus control
0B4	Local bus address
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

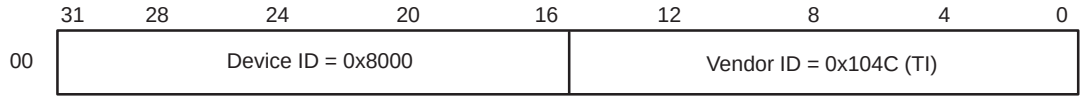
DMA CONTROLLER REGISTERS	
100	DMA channel 0 previous packet control list address/temp
104	DMA channel 0 current packet control list address
108	DMA channel 0 current data buffer address
10C	DMA channel 0 status
110	DMA channel 0 control
114	DMA channel 0 ready
118	DMA channel 0 current state
120	DMA 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
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
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
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
198	DMA channel 4 current state
1A0 to 8E0	This range of address offsets is reserved for future use in implementing status and control registers for DMA channels 5 through 63
900	DMA diagnostic test control
904	DMA receive FIFO packet count
908	DMA global
FIFO REGISTERS	
A00	FIFO size
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
A14	ATF-ITF transmit data ready threshold control
A20	General receive FIFO pop-push port 0 FIFO bit 32 set to 0 on write
A24	General receive FIFO pop-push port 1 FIFO bit 32 set to 1 on write
A28	Not used
A2C	Not used
A30	Asynchronous transmit FIFO pop-push port 0 FIFO bit 32 set to 0 on write
A34	Asynchronous transmit FIFO pop-push port 1 FIFO bit 32 set to 1 on write
A38	Not used
A3C	Not used
A40	Isochronous transmit FIFO pop-push port 0 FIFO bit 32 set to 0 on write
A44	Isochronous transmit FIFO pop-push port 1 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
B04	DMA channel 0 word 0 receive packet comparator mask
B08	DMA channel 0 word 1 receive packet comparator value
B0C	DMA channel 0 word 1 receive packet comparator mask
B10	DMA channel 1 word 0 receive packet comparator value
B14	DMA channel 1 word 0 receive packet comparator mask
B18	DMA channel 1 word 1 receive packet comparator value
B1C	DMA channel 1 word 1 receive packet comparator mask
B20	DMA channel 2 word 0 receive packet comparator value
B24	DMA channel 2 word 0 receive packet comparator mask
B28	DMA channel 2 word 1 receive packet comparator value
B2C	DMA channel 2 word 1 receive packet comparator mask
B30	DMA channel 3 word 0 receive packet comparator value
B34	DMA channel 3 word 0 receive packet comparator mask
B38	DMA channel 3 word 1 receive packet comparator value
B3C	DMA channel 3 word 1 receive packet comparator mask
B40	DMA channel 4 word 0 receive packet comparator value
B44	DMA channel 4 word 0 receive packet comparator mask
B48	DMA channel 4 word 1 receive packet comparator value
B4C	DMA channel 4 word 1 receive packet comparator mask
B50 to EF0	This range of address offsets is reserved for future use in implementing comparator control registers for DMA channels 5 through 63
F00	1394 bus number – node number
F04	1394 link layer control
F08	1394 cycle timer
F0C	1394 physical layer access control
F10	1394 diagnostic test control
F14	1394 link layer interrupt status
F18	1394 link layer interrupt enable
F1C	1394 busy retry count and retry interval
F20	LLC state machine vector monitor port 1
F24	FIFO overrun-underrun error counters

5.2 PCI Configuration and Miscellaneous Register Definitions

5.2.1 Device-Vendor ID @000

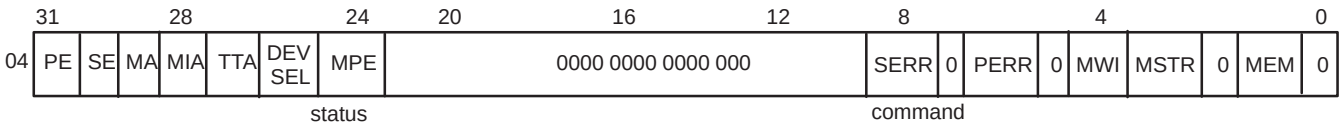
This register provides application software access to the vendor ID and device ID numbers that are assigned to the PCILynx ASIC. This register is read-only.



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

5.2.2 Command – Status @004

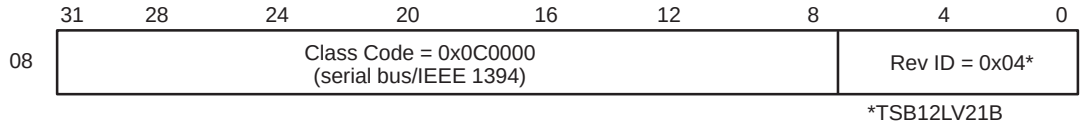
This register provides application software with an interface for controlling the PCILynx PCI operating behavior and monitoring the PCI slave and master operation status. This register is initialized to 0x02000000 on power-up reset. Status bits 31–16 are cleared by writing a 1 to the bit position to be cleared.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31	PARERR	r/w	PCILynx detected a parity error. (parity_detected = 1, no_error = 0)
30	SYSERR	r/w	PCILynx asserted SERR# signal. (asserted = 1, not_asserted = 0)
29	MSTABT	r/w	PCILynx as master aborted PCI transaction. abort = 1
28	MST_TGTABT	r/w	PCILynx as master was aborted by the target. abort = 1
27	TGT_TGTABT	r/w	PCILynx asserted target abort as a target. abort = 1
26–25	DEVSEL[1–0]	r	Devsel# timing setting. 01=medium read-only
24	MSTR_PAR	r/w	PCILynx as master detected 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	Return 0s 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 (MSTR_PAR), 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	PCILynx 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

5.2.3 Class Code – Revision ID @008

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

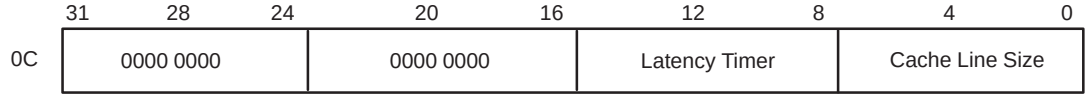


BIT NO.	BIT NAME	DIR	DESCRIPTION	
31–08	CLASS_CD[23–0]	r	PCI class code = 0x0C0000 (serial bus / 1394) (fixed)	read-only
07–00	REV_ID[7–0]	r	PCI revision code. (fixed)	read-only

Revision ID: 0x00 = TSB12LV21PGF (production) [f643950],
0x01 = TSB12LV21APGF (prototype) [f643178]
0x02 = TSB12LV21APGF (production) [f643178A],
0x03 = Custom device
0x04 = TSB12LV21BPGF (production) [f711327]

5.2.4 Header Type-Latency TImer-Cache LIne Size @00C

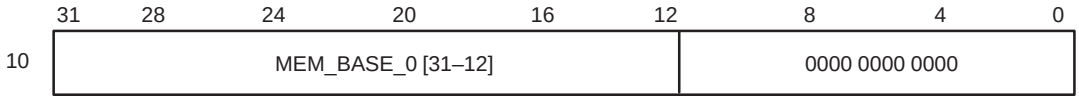
This register provides an interface for application software to program the PCI cache line size, PCI latency timer, PCI header type, and PCI built-in test parameters. This register is set to 0x00000000 on power-up reset.



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	

5.2.5 Memory Access Base Address 0 – PCILynx Internal Registers @010

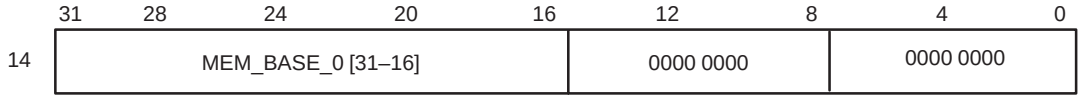
This register provides the interface for application software to set the memory access base address of the internal PCILynx register set. This register is set to 0x00000000 on power-up reset (or 0x00010000 on power-up reset if autoboot is enabled).



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

5.2.6 Memory Access Base Address 1 – External RAM Port @014

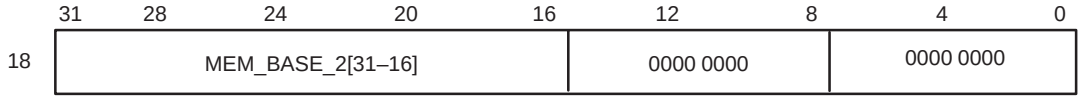
This register provides the interface for application software to set the memory access base address of the external RAM attached to the LYNX local bus. This register is set to 0x00000000 on power-up reset.



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

5.2.7 Memory Access Base Address 2 – AUX Port @018

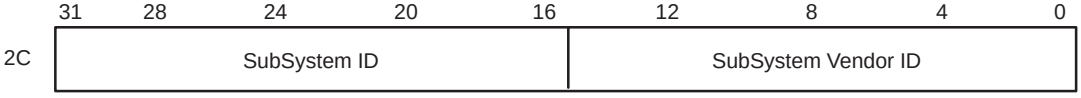
This register provides the interface for application software to set the memory access base address of the AUX port on the LYNX local bus. This register is set to 0x00000000 on power-up reset. The ZV port occupies upper 4K (0xF000–0xFFFF) of the AUX port address space when CLK is set to a valid clock 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

5.2.8 Subsystem ID @02C

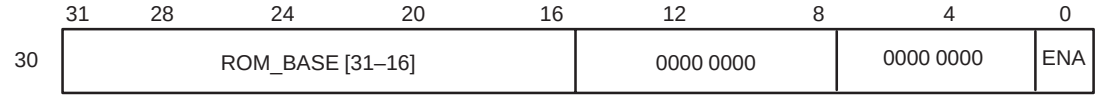
This register provides a method for the subsystem vendor to uniquely identify his device. The subsystem vendor ID is assigned by the PCI SIG to ensure uniqueness. This register is set to 0x00000000 on power-up reset and then these ID's are loaded from the serial EEPROM. Also see Appendix F – *Serial EEPROM Data*.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–16	Subsystem_ID[15–0]	r	Unique subsystem ID (initialized from serial EEPROM)
15–00	Subsystem vendor ID[15–0]	r	Unique subsystem vendor ID (initialized from serial EEPROM)

5.2.9 Expansion ROM Base Address @030

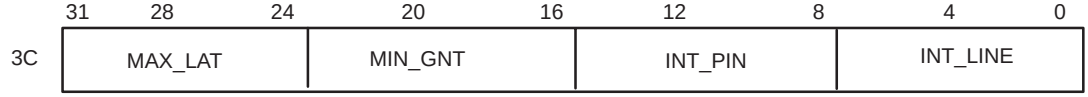
This register provides an interface for application software to set the memory access base address of the PCILynx external expansion ROM. To access this address space, both the base address should be set to an appropriate PCI address and the ROM enable bit (bit 0) set to 1. This register is set to 0x00000000 on power-up reset when AUTOBOOT is inactive and this register is set to 0x00000001 on power up when AUTOBOOT is active.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–16	ROMBASE[31–16]	r/w	PCILynx expansion ROM base address
15–01	Reserved	r	Return 0s when read.
00	ROMEN	r/w	Enable expansion ROM access. (enable = 1, disable = 0)

5.2.10 Max_Latency–Min_Grant–Int__Pin–Int_Line Register @03C

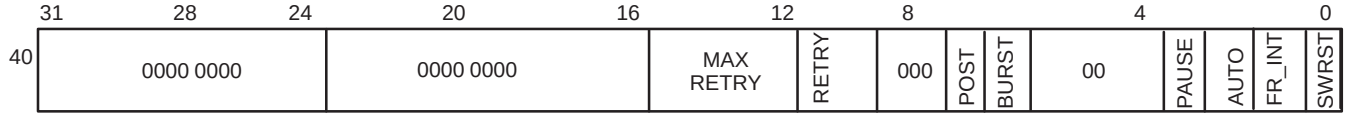
This register provides the interface for application software to read values for the Max_Latency, Min_Grant, and interrupt pin parameters, and to write/read values for the interrupt line. The interrupt line register is set to 0x00 on power-up reset. The interrupt pin register is read-only and fixed at a value of 0x01. The minimum grant is set to 0x01, and maximum latency is set to 0x02 on power reset. After power reset, the MAX_LAT and MIN_GNT registers are loaded from serial EEPROM (if present). The PCI interface returns retry status to any accesses while the serial EEPROM machine is active initializing these locations. Also see Appendix F – *Serial EEPROM Data*.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–24	MAX_LAT[7–0]	r	Maximum latency (reset to 0x02, then initialized from serial EEPROM)
23–16	MIN_GNT[7–0]	r	Minimum grant time (reset to 0x01, then initialized from serial EEPROM)
15–08	INT_PIN[7–0]	r	Interrupt pin used. INTPIN = 0x01 = $\overline{\text{INTA}}$
07–00	INT_LINE[7–0]	r/w	Interrupt line – indicates which interrupt PCILynx is connected to (set by system software)

5.2.11 Miscellaneous Control @040

This register provides an interface for application software to perform miscellaneous control operations. This register also supplies operational status information. This register is 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 PCILynx master will attempt when a retry termination status occurs. (0 = infinite number of retries)
11	ENA_MST_RTY	r/w	Enable PCILynx master cycle retry count. (enable = 1, disable = 0)
10–08	Reserved	r	Return 0s when read.
07	ENA_POST_WR	r/w	Enable PCI slave posted writes. Operational software should set to 0 in PCILynx Rev A and above. (enable = 1, disable = 0)
06	ENA_SLV_BURST	r/w	Enable PCI slave burst. Operational software should set to 0 in PCILynx Rev A and above to ensure PCI Bus Specification, Revision 2.1 compliance. (enable = 1, disable = 0)
05–04	Reserved	r	Return 0s when 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 input pin.
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.

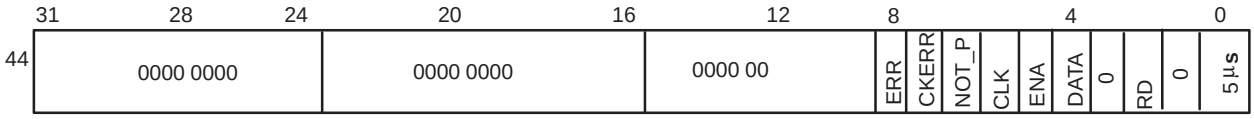
5.2.12 Serial EEPROM Control @044

This register provides an interface for application software to control the read of the PCILynx external serial EEPROM. This register is 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-µsec 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 5 µsec after starting the timer.

> **NOTE:** Whenever the EEPROM is driving EEPDAT and the TIMER_5USEC pin is to be set, simply logical ORing the TIMER_5USEC bit is not appropriate since performing a read of this register when the EEPROM is just starting to drive EEPDAT may produce either a logic 0 or 1 depending on the speed of the EEPROM and CPU. It is better to write TIMER_5USEC ORed with values of EEPCLK and EEPDATA as determined to be current values if using the TIMER_5USEC inside serial EEPROM programming protocol.

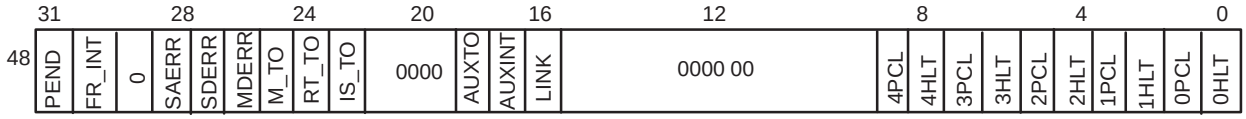
The 5-µsec timer is derived from the phy_clk50 clock from the 1394 physical layer device since the PCI clock frequency is not guaranteed. This timer will not function if the 1394 physical layer device is not present or is not providing a clock output. Care should be used in writing software for this timer to avoid software hang conditions in the event of phy_clk50 not running. For example, the system clock could be used to escape from 5-µsec timer polling after 1 second has expired on the system clock.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–10	Reserved	r	Return 0s when 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. (only if EEPRNA = 1) high = 1, low = 0 Read: read value of serial EEPROM clock signal
05	EEPENA	r/w	Select serial EEPROM interface controlling outputs. Serial EEPROM control register controls outputs = 1 PCILynx internal SEEPROM state machine controls outputs = 0
04	EEPDAT	r/w	Write: Output serial EEPROM data. (only if EEPRNA = 1) (high = 1, low = 0) Read: read value of serial EEPROM data signal
03	Reserved	r	Returns 0 when read.
02	EEPSTARTRD	w	Restarts serial EEPROM read state machine
01	Reserved	r	Returns 0 when read.
00	TIMER_5USEC	r/w	5-µsec timer. (Time expired = 1, Start timer = 0)

5.2.13 PCI Interrupt Status @048

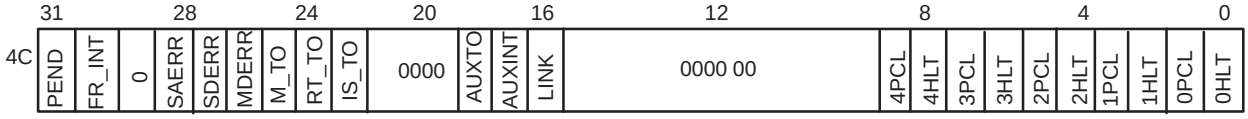
This register provides the interface for application software to determine the events which generate an $\overline{\text{INTA}}$ interrupt. This register is set to 0x00000000 on power-up reset. Interrupt status is cleared by writing a 1 to the bit to be cleared; the interrupt status bit is cleared only 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	Returns 0 when 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	Return 0s when read.
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
15–10	Reserved	r	Return 0s 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

5.2.14 PCI Interrupt Enable @04C

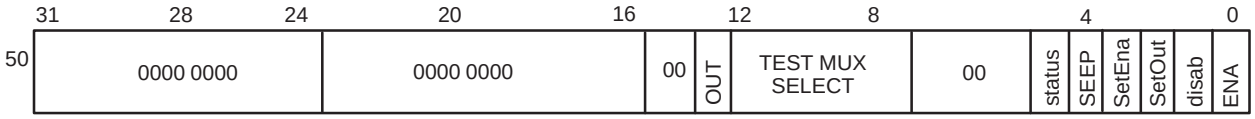
This register provides an interface for application software to selectively enable the events which can cause an interrupt to occur. This register is set to 0x00000000 on power-up reset.



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	Return 0s when read.
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 register.
15–10	Reserved	r	Return 0s 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

5.2.15 PCI Test Register @050

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

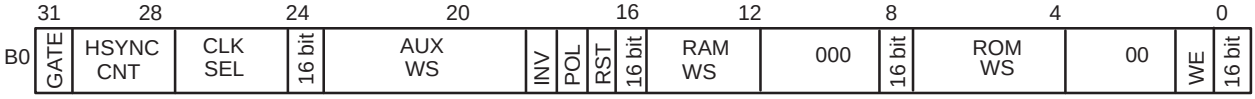


BIT NO.	BIT NAME	DIR	DESCRIPTION	
31–14	Reserved	r	Return 0s when read.	
13	TEST_OUTPUT	r	Test output from test mux.	
12–08	TEST_MUX_SEL	r/w	Test mux select.	
07–06	Reserved	r	Return 0s when read.	
05	TEST_STATUS	r/w	Interrupt status register test mode (R/W).	Test mode = 1, normal = 0
04	TEST_SEEPROM	r/w	Serial EEPROM test mode.	Test mode = 1, normal = 0
03	SET_OUTPUT_EN	r/w	Set output enables.	set = 1, normal = 0
02	SET_OUTPUT_FF	r/w	Set output flip-flops.	set = 1, normal = 0
01	DISABLE_DRIVERS	r/w	3-states normally output-only drivers.	3-stated = 1, normal = 0
00	TEST_REG_EN	r/w	Test register enable.	enable = 1, normal = 0

TEST_MUX_SELECT [4–0]	SELECTED SIGNAL	TEST_MUX_SELECT [4–0]	SELECTED SIGNAL
0x00	test_nand_chain output	0x10	PCI module – internal_mstr_act
0x01	dma_test_mux output	0x11	PCI module – pci_mstr_st
0x02	fifo_test_mux output	0x12	PCI module – aux_pci_act
0x03	link_test_mux output	0x13	PCI module – m_addr
0x04	PCI module – mstr_act	0x14	PCI module – m_data
0x05	PCI module – mstr_err	0x15	PCI module – pci_mstr_xfrq
0x06	PCI module – mstr_req	0x16	PCI module – mstr_byte_cnt_eq_0
0x07	PCI module – mstr_xfr	0x17	PCI module – pci_xfr_cnt_eq_0
0x08	PCI module – mstr_ack	0x18	PCI module – in_buf_full
0x09	PCI module – mstr_internal_cyc	0x19	PCI module – wr_buf_full
0x0A	PCI module – my_slv_cyc	0x1A	PCI module – wr_buf_empty
0x0B	PCI module – slv_data	0x1B	PCI module – mstr_fifo_rdy
0x0C	PCI module – slv_rd	0x1C	PCI module – aux_busy
0x0D	PCI module – slv_wr	0x1D	PCI module – zv_busy
0x0E	PCI module – int_slv_xfr	0x1E	PCI module – retry_slv
0x0F	PCI module – int_pci_slv_act	0x1F	PCI module – reserved

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

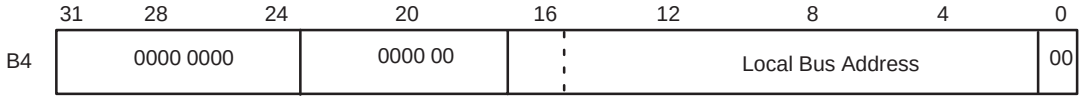
This register provides an interface for application software to control the pacing of data transferred on the local bus to/from attached external devices. This register also specifies the data bus width required for each external device type and the polarity of incoming interrupts. This register is initialized to 0x000000E0 on PCI reset and then loaded from SEEPROM. Each byte controls a different area – ROM, RAM, AUX and ZV. In early devices (before PCILynx Rev A), GPIO polarity control does not work as documented here. Also see Appendix F – *Serial EEPROM Data*.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31	GATE_PIXEL_CLK	r/w	ZV pixel clock gating enable. 0 = free running pixel clock; zv_data_valid signal must be used to determine when valid ZV data is present. 1 = gating enabled (gated mode); zv_pix_clk only toggles when valid ZV data is present.
30–28	HSYNC_CNT[2–0]	r/w	Horizontal sync count. Number of horizontal sync decodes (i.e., writes to zoom port at address 0xF004) required for each horizontal sync actually asserted on the ZOOM port HSYNC signal. Useful for formats having multiple packets per horizontal line. 0 0 0 hsync cnt = 0. A hsync will still be generated for every frame 0 0 1 hsync cnt = 1 0 1 0 hsync cnt = 2 0 1 1 hsync cnt = 4 1 0 0 hsync cnt = 6 1 0 1 hsync cnt = 8 1 1 0 hsync cnt = 10 1 1 1 hsync cnt = 12
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, all others are 16-bit mode only. AUX address space F000–FFFF is allocated to ZV (ZV is enabled) when one of the six 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	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 waitstates to insert for external AUX access. A value of 0xF causes waitstates to be inserted until either the AUX_RDYz input pin is asserted or 0xF waitstates have occurred.
19	INVERT_ZV_CLK	w	1 = invert, 0 = don't invert
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	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 waitstates to insert for external RAM access. A value of 0xF causes waitstates to be inserted until either the $\overline{\text{AUX\_RDY}}$ input pin is asserted or 0xF waitstates have occurred.
11–09	Reserved	r	Return 0s when read.
08	RAM_16	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 waitstates to insert for external ROM access. A value of 0xF causes waitstates to be inserted until either the $\overline{\text{AUX\_RDY}}$ input pin is asserted or 0xF waitstates have occurred.
03–02	Reserved	r	Return 0s when read.
01	ROM_WR_EN	r/w	ROM write enable (writable nonvolatile memory)
00	ROM_16	r/w	Data width, 1 = ROM access is 16 bits wide, 0 = ROM access is 8 bits wide

5.2.17 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 autoincrements every time it is used. This address must be specifically written to reinitialize. This register is initialized to 0x00000000 on power-up reset.

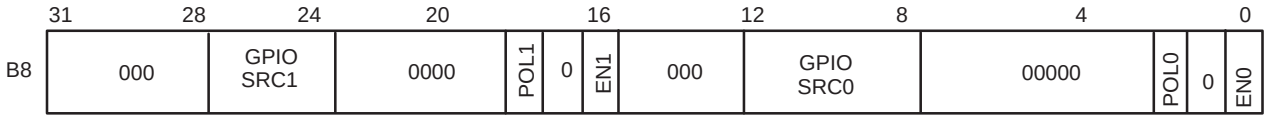


17	16	Local Bus Select
0	0	RAM_SEL
0	1	ROM_SEL
1	0	AUX_SEL
1	1	ZV_SSEL

BIT NO.	BIT NAME	DIR	DESCRIPTION
31–18	Reserved	r	Return 0s when read.
17–16	ADDR_SELECT[1–0]	r/w	00 = SRAM, 01 = ROM, 10 = AUX, 11 = ZOOM
15–02	AUX_ADR[15–02]	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 rewritten to reinitialize.
01–00	Reserved	r	Return 0s when read.

5.2.18 PCI_GPIO[1–0] Control Register A @0B8

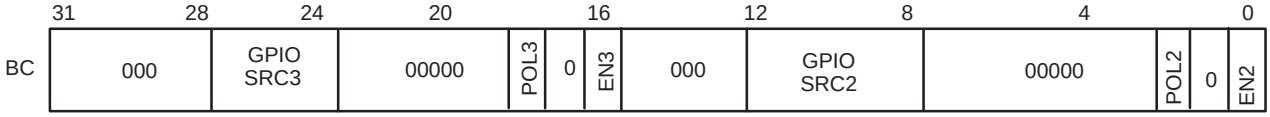
This register provides application software with an interface for configuring the operating mode of GPIO[1–0] port pins. This register is initialized to 0x00000000 on power-up reset. Each 16-bit register half is accessed separately (i.e., 8- or 16-bit write).



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–29	Reserved	r	Return 0s when read.
28–24	GPIO_SRC1[4–0]	r/w	Data bit mux select for output on GPIO[1]
23–19	Reserved	r	Return 0s when read.
18	GPIO_POL1	r/w	GPIO[1] output polarity control (0 = noninverted, 1 = inverted)
17	Reserved	r	Returns 0 when read.
16	GPIO_OUT_EN1	r/w	GPIO[1] output enable control (0 = 3-state, 1 = enabled)
15–13	Reserved	r	Return 0s when read.
12–08	GPIO_SRC0[4–0]	r/w	Data bit mux select for output on GPIO[0]
07–03	Reserved	r	Return 0s when read.
02	GPIO_POL0	r/w	GPIO[0] output polarity control (0 = noninverted, 1 = inverted)
01	Reserved	r	Returns 0 when read.
00	GPIO_OUT_EN0	r/w	GPIO[0] output enable control (0 = 3-state, 1 = enabled)

5.2.19 PCI_GPIO[3–2] Control Register B @0BC

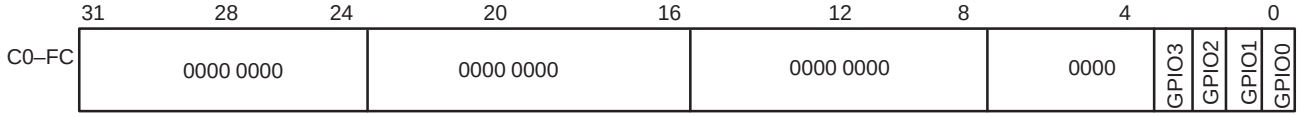
This register provides application software with an interface for configuring the operating mode of GPIO[3–2] port pins. This register is initialized to 0x00000000 on power-up reset. Each 16-bit register half is accessed separately (i.e., 8- or 16-bit write).



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–29	Reserved	r	Return 0s when read.
28–24	GPIO_SRC3[4–0]	r/w	Data bit mux select for output on GPIO[3]
23–19	Reserved	r	Return 0s when read.
18	GPIO_POL_OUT3	r/w	GPIO[3] output polarity control (0 = noninverted, 1 = inverted)
17	Reserved	r	Returns 0 when read.
16	GPIO_OUT_EN3	r/w	GPIO[3] output enable control (0 = 3-state, 1 = enabled)
15–13	Reserved	r	Return 0s when read.
12–08	GPIO_SRC2[4–0]	r/w	Data bit mux select for output on GPIO[2]
07–03	Reserved	r	Return 0s when read.
02	GPIO_POL_OUT2	r/w	GPIO[2] output polarity control (0 = noninverted, 1 = inverted)
01	Reserved	r	Returns 0 when read.
00	GPIO_OUT_EN2	r/w	GPIO[2] output enable control (0 = 3-state, 1 = enabled)

5.2.20 PCI GPIO DATA Read-Write Ports @0C0 through @0FC

This register provides application software with an interface for reading and writing the four general purpose input/output ports, GPIO[3–0].



The PCI address offsets indicated in the following table, are 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 determines 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. This allows the programmer to set a 4-bit mask of the GPIO ports to be considered, shift this mask left 2, and use the result to add as an offset to 0xC0. For example, to write to only GPIO port 0, use offset 0xC0+ (0x01<<2) = 0xC4. To write to GPIO port 3 only, use offset 0xC0 + (0x08<<2) = 0xE0.

PCI slave writes to these address offsets must write 32 bits to write to any GPIO port.

The data bit value that is written to a GPIO port is 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, is 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.

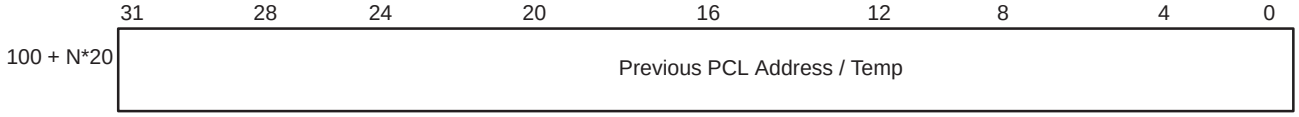
A read of the GPIO register always returns the value read from all four GPIO ports.

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]

5.3 DMA Control and Status Register Definitions

5.3.1 DMA Channel 0 through 4 – Previous Packet Control List Address/Temp @100, 120, 140, 160, 180

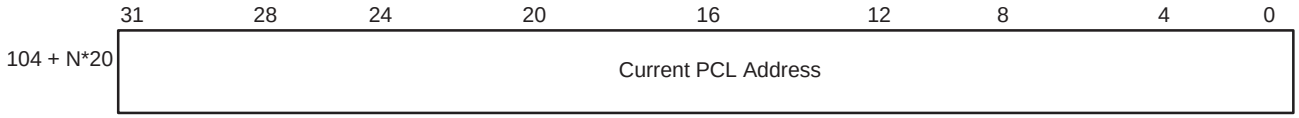
This register contains the address of the previous packet control list which is being processed by the active DMA channel when programmed for asynchronous transmit operations. This register is also used during the execution of auxiliary commands to temporarily hold data during a load or store command.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



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.

5.3.2 DMA Channel 0 through 4 – Current Packet Control List Address @104, 124, 144, 164, 184

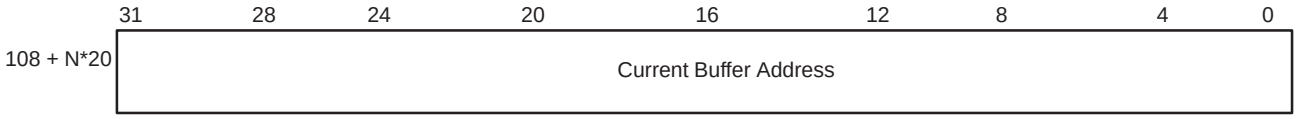
This register specifies the address of the current packet control list which is being processed by the active DMA channel. This register is initialized by application software to point to a PCL with a valid next address pointing to the start of the first packet control list, which is the first in a queue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

5.3.3 DMA Channel 0 through 4 – Current Data Buffer Address @108, 128, 148, 168, 188

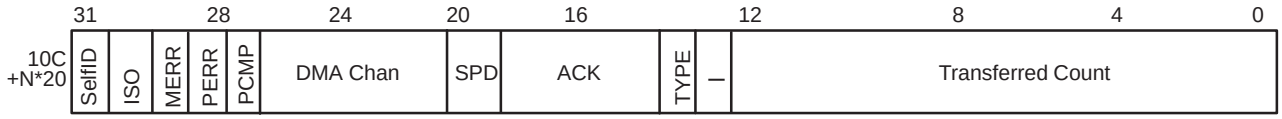
This register contains the start address of the host memory data buffer that is being processed by the active DMA channel. The active DMA channel loads this register with the data buffer start address that is obtained from the current packet control list. This register is used by the DMA to read in the next PCL address for validation. As a result, it may change during a PCL link operation. This register is initialized to 0x00000000 on power-up reset. The contents of this register are lost when the DMA chains to another PCL as this register is used to evaluate the validity of the next PCL pointer.



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.

5.3.4 DMA Channel 0 through 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 stores status from this register back at packet control list offset 0xC. This register is initialized to 0x000000000 on power-up reset. This is roughly the same information as appears 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.



STATUS		
BIT NO.	**BIT NAME**	**DESCRIPTION**
31	Self ID	Set when a self ID packet has been received by this channel. Refer to the definition of the receive comparator registers for how a channel is enabled for self ID reception.
30	ISO MODE	The received packet was an isochronous packet.
29	Mst Err	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 writable with the opposite of write data when DMATESTEN is 1.
28	Pkt Err	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.
27	Pkt Cmp	Packet complete. Written by the DMA upon completion of this packet.
26–21	Receive Dma_Cha[5–0]	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 return 0s for other commands.

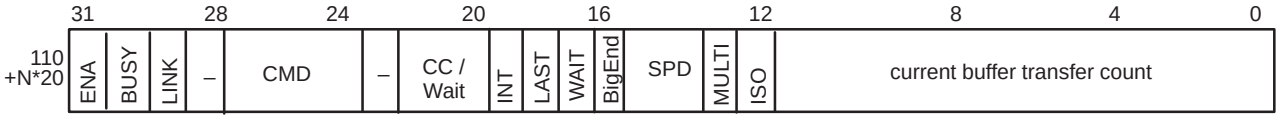
Dma_Cha[5–0]	DMA Channel Number
0 0 0 0 0 0	0
0 0 0 0 0 1	1
0 0 0 0 1 0	2
0 0 0 0 1 1	3
0 0 0 1 0 0 Others	4 Reserved

20–19	Rcv_Speed[1–0]	The speed at which the packet was received for asynchronous or isochronous transfers. Valid only for channels programmed for receive operations. These bits return 0s for other commands. 00 = 100 Mbps 01 = 200 Mbps 10 = 400 Mbps

STATUS (Continued)		
BIT NO.	**BIT NAME**	**DESCRIPTION**
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 0s 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 an ACK_TIMEOUT 0010 = Link reported a FIFO underrun 0011 = Link reported a CRC error on a received 1394 ack packet 0100 = DMA received an end of packet token while expecting a start of packet token. Catastrophic internal error. 0101 = No expected end of receive packet 0110 = Pipelined asynchronous transmit command encountered a command other than another asynchronous 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 (header + payload) for this packet. These bits are indeterminate for asynchronous transmits due to the potentially pipelined nature of asynchronous XMT commands. The count will also include any retried packets during asynchronous receives. These bits are written with 0s after completion of auxiliary commands.

5.3.5 DMA Channel 0 through 4 – DMA Channel Control @110, 130, 150, 170, 190

This register provides the interface for application software to initiate the operation a DMA channel and to monitor its operational status. This register is 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 the PCL's control. The PCL should be consulted instead if the DMA is still running since this register can be changing as the DMA processes multiple PCLs.



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 immediately.
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 refetch the next address entry of the current PCL and perform a check on the valid bit. If the valid bit is set, then 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	This bit is ignored when read and written with 0.
27–24	CMD3–0	r	Command select. These bits control what command the DMA channel will execute.

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	SWAP & COMPARE (PCILynx Rev A and higher)

23	Reserved	r	This bit is ignored when read and written with 0.

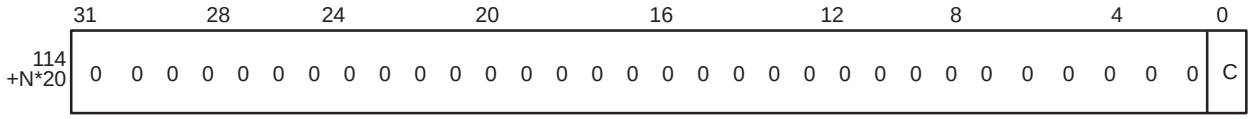
<table>
<tr><th>BIT NO.</th><th>BIT NAME</th><th>DIR</th><th>DESCRIPTION</th></tr>
<tr><td>22–20</td><td>Condition codes 2–0
(branch command)</td><td>r</td><td>Branch command condition codes. These bits select what conditions have to be met during the execution of the branch command to cause the address contained in destination to be loaded into the next PCL address and linked.
<table>
<tr><th>Condition code [2–0]</th><th>Branch condition</th></tr>
<tr><td>000</td><td>Don’t branch</td></tr>
<tr><td>001</td><td>Branch if DMA ready register = 1 (this channel)</td></tr>
<tr><td>010</td><td>Branch if DMA ready register = 0 (this channel)</td></tr>
<tr><td>011</td><td>Branch if external ready pin RDY = 1 (this channel)</td></tr>
<tr><td>100</td><td>Branch if external ready pin RDY = 0 (this channel)</td></tr>
<tr><td>101</td><td>Branch if GPIO port 2 is active (this channel)</td></tr>
<tr><td>110</td><td>Branch if GPIO port 3 is active (this channel)</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.
<table>
<tr><th>Wait select [2–0]</th><th>Wait condition</th></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 channel)</td></tr>
<tr><td>010</td><td>Wait for DMA ready register = 0 (this channel)</td></tr>
<tr><td>011</td><td>Wait for external ready pin RDY = 1 (this channel)</td></tr>
<tr><td>100</td><td>Wait for external ready pin RDY = 0 (this channel)</td></tr>
<tr><td>101</td><td>Wait for GPIO port 2 to go active (this channel)</td></tr>
<tr><td>110</td><td>Wait for GPIO port 3 to go active (this channel)</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)</td></tr>
<tr><td>18</td><td>LAST BUFF</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.</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)</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] = 00–100 Mbps
xmt_spd_code[1–0] = 01–200 Mbps
xmt_spd_code[1–0] = 10–400 Mbps
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>r</td><td>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 isochronous transmit channels so all isochronous channels should set the multi-isochronous bit set to the same value to prevent otherwise unpredictable behavior.</td></tr>
</table>

BIT NO.	BIT NAME	DIR	DESCRIPTION
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. Also must be written (1 or 0) whenever this channel's link bit is set. This is required to insure proper fairness of isochronous XMT's if more than one DMA channel is to be used for isochronous XMT's.
11–00	DBXXFRLEN[11–0]	r	Remaining count to be transferred of the current buffer. Loaded by the DMA from the PCL.

5.3.6 DMA Channel 0 through 4 – DMA Ready Register @114, 134, 154, 174, 194

This register is implemented to provide a mechanism for pacing the DMA. The wait select bits in the PCL data buf0 ctl/byte_cnt/cmd 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.

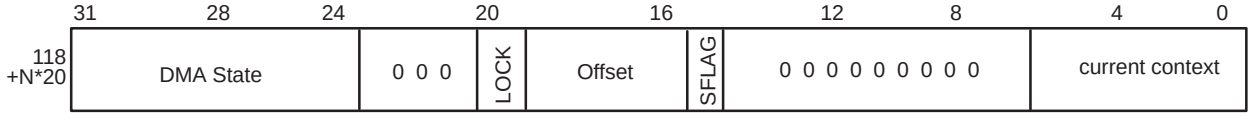
The auxiliary branch command can also use this register to conditionally branch. The condition select bits in the PCL data buf0 ctl/byte_cnt/cmd can select the contents of register to be used in a decision to branch to another PCL. This register is set to 0x000000000 on power-up reset.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–01	Unused	r/w	Return 0s when read, ignored when written
00	CONDITION	r/w	This bit is used for wait or branch conditional checks.

5.3.7 DMA Channel 0 through 4 – Current DMA State @118, 138, 158, 178, 198

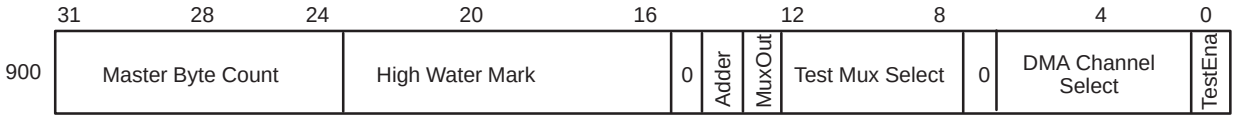
This register provides an interface for application software to read the state_vector of a DMA channel. The active DMA channel uses this register to store its state vector and other flag bits used during its active or idle period. This register is intended for debug purposes only.



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	Return 0s 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	Return 0s when read.
05–00	CURRENT CONTEXT	r	Current channel context selected by the priority encoder and executing by the state machine.

5.3.8 DMA Diagnostic Test Control @900

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



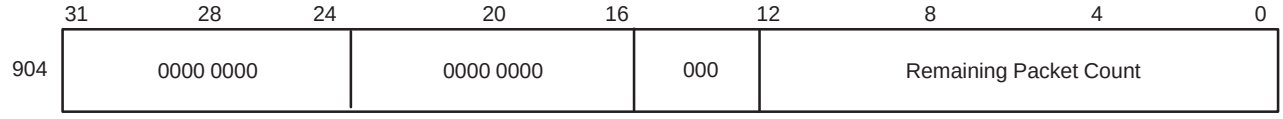
BIT NO.	BIT NAME	DIR	DESCRIPTION
31–24	MASTER BYTE COUNT	r	This is the computed number of bytes that the DMA will request during a PCI master cycle. The master byte count is equal to the lesser of the current HIGH WATER MARK–4, the current receive transfer count, and the DBXXFRLEN bits of the DMA channel control word. The selected channel is determined by the channel select bits of this register.
23–16	HIGH WATER MARK	r	This is the computed FIFO threshold which must be met before the DMA will request a PCI master cycle. It is equal to high water mark which is equal to the (lower bound field of the DMA global register * 8) + 4. (PCILynx Rev A and higher)
15	0	r	Returns 0 when read.
14	ADDERTEST	r/w	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 – 4, the receive transfer count register, and the DBXXFRLEN bits of the DMA channel control register.
13	DMA Test Mux Select Out	r	Read back of the signal selected by the above DMA test mux.

<table>
<tr><th>BIT NO.</th><th>BIT NAME</th><th>DIR</th><th>DESCRIPTION</th></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>async_xmt_flush *</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>1</td><td>rcv_start *</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>rcv_end *</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>pending_active_context_switch *</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>0</td><td>pending context_switch *</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>1</td><td>xmt_ack_type *</td></tr>
<tr><td></td><td></td><td>:</td><td></td><td></td><td>rcv_link_active</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>rcv_link_active</td></tr>
</table></td></tr>
<tr><td>07</td><td>Unused</td><td>r</td><td>Returns 0 when read.</td></tr>
<tr><td>06–01</td><td>CHANNEL SELECT[5–0]</td><td>r/w</td><td>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.
<table>
<tr><th>CHANNEL SELECT[5–0]</th><th>DMA Channel Number</th></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>:</td><td>:</td></tr>
<tr><td>0 0 0 1 0 0</td><td>4</td></tr>
<tr><td>others</td><td>Reserved</td></tr>
</table></td></tr>
<tr><td>00</td><td>DMATESTEN</td><td>r/w</td><td>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>

* PCILynx Rev A and higher

5.3.9 Receive Packet Remaining Count Register @904

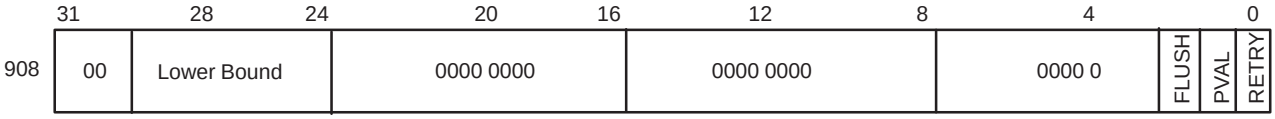
This register contains the current received packet count. The DMA loads this register with the receive packet count passed in the receive FIFO token words. This count is then decremented as the data is transferred to the PCI bus.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–13	Unused	r	Return 0s when read.
12–00	REMAINING PACKET COUNT	r	The current remaining packet count.

5.3.10 Global Register @908

This register contains the transfer threshold, some flags, and miscellaneous information that the DMA uses during an asynchronous transmit operation.

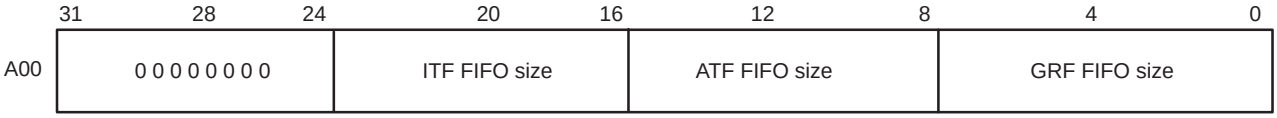


BIT NO.	BIT NAME	DIR	DESCRIPTION
31–29	Unused	r	Return 0s when read.
28–24	LOWER BOUND	r/w	This is the threshold (from the DMA's PCI point of view) which must be reached before a transfer will be requested of the PCI bus. For receives it is the amount of data which must be in the GRF (if less than a full packet) before a PCI write will be requested of the PCI bus. For transmits it is the amount of room available in the ITF or ATF before a read is requested of the PCI bus. The value represented here is the number of double quadlets (8 bytes) plus 4. For example: 00 = 4-byte threshold 01 = 12-byte threshold 02 = 20-byte threshold : 1F = 252-byte threshold These bits will be set to 0x2 upon reset. Once the threshold has been reached, a request equal to the lesser of the cache line size register or the lower bound register – 4 is made of the PCI bus.
23–03	Unused	r	Return 0s when read.
02	FIFO FLUSH	r	Request the link to flush the asynchronous transmit FIFO.
01	PREVIOUS VALID	r	A PCL is currently pipelined and awaiting a status update.
00	RETRY	r	An asynchronous transmit retry is in progress.

5.4 FIFO Control and Status Register Definitions

5.4.1 FIFO Size @A00

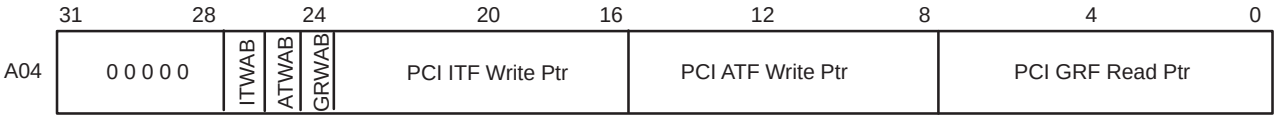
This register provides the application software with an interface for programming the size of each of the logical FIFOs described in section 4.2.3, *FIFO Logic*. Each FIFO is programmable in size from 0 to 255. Each decimal value represents 4 quadlets (16 bytes). For example, 2 = 8 quadlets or 32 bytes, 10 = 40 quadlets or 160 bytes, etc. For any given combination, the sum of all 3 FIFO sizes is less than or equal to 256 (1024 quadlets). This register is initialized to 0x00000000 on power-up reset. The minimum usable FIFO size is equal to the cache line size register times 2. The sizing of the FIFO depends on the application and PCI latencies.



BIT NO.	BIT NAME	DIR	DESCRIPTION		
31–24	Not used	r	Return 0s when read.		
23–16	ITF_FIFOSZ[7–0]	r/w	Isochronous transmit FIFO size	0x00 <= size <= 0xFF	(X 4 quadlets)
15–08	ATF_FIFOSZ[7–0]	r/w	Asynchronous transmit FIFO size	0x00 <= size <= 0xFF	(X 4 quadlets)
07–00	GRF_FIFOSZ[7–0]	r/w	General receive FIFO size	0x00 <= size <= 0xFF	(X 4 quadlets)

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

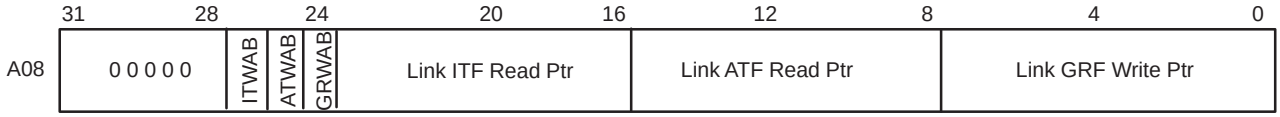
This register is implemented to provide the application software with a PCI slave access port for writing to or reading from the FIFO pointers which are used in accessing the PCI-side of the FIFO.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–27	Not used	r	Return 0s when read.
26	ITF_WAB_L	r/w	PCI-side ITF write pointer wraparound bit
25	ATF_WAB_L	r/w	PCI-side ATF write pointer wraparound bit
24	GRF_WAB_L	r/w	PCI-side GRF read pointer wraparound bit
23–16	PCI_ITF_WPTR[7–0]	r/w	PCI-side isochronous transmit FIFO write pointer value returned on read = itf pointer contents
15–08	PCI_ATF_WPTR[7–0]	r/w	PCI-side asynchronous transmit FIFO write pointer value returned on read = atf pointer contents + (ITF_size)
07–00	PCI_GRF_RPTR[7–0]	r/w	PCI-side general receive FIFO read pointer. value returned on read = grf pointer contents + (ATF_size + ITF_size)

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

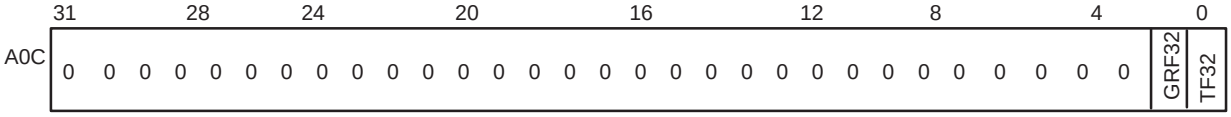
This register provides the application software with a PCI slave access port for writing to or reading from the FIFO pointers which are used in accessing the link-side of the FIFO.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–27	Not used	r	Return 0s when read.
26	ITF_WAB_L	r/w	Link-side ITF read pointer wraparound bit
25	ATF_WAB_L	r/w	Link-side ATF read pointer wraparound bit
24	GRF_WAB_L	r/w	Link-side GRF write pointer wraparound 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)

5.4.4 FIFO Control Token Status Read-Port @A0C

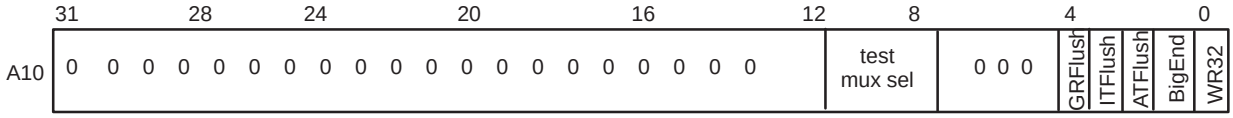
This port provides an interface for application software to obtain the value of bit 32 (MSB) of the 33-bit data value from the last FIFO that was popped.



BIT NO.	BIT NAME	Function
31–02	Not used	Return 0s when 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

5.4.5 FIFO Control and Test Register @A10

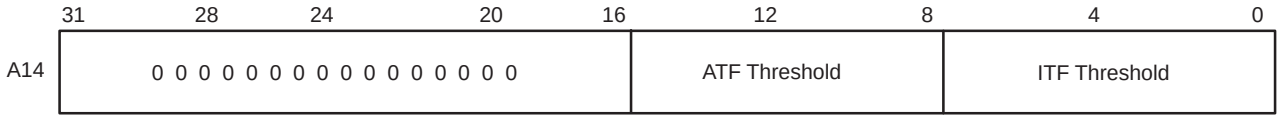
This register provides the application software with an interface for test control and flushing of a selected FIFO. This register is initialized to 0x00000000 on power-up reset.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–10	not used	r	Return 0s when read.
11–08	TEST_MUX[3–0]	r/w	Selects 1 of 16 test points in the FIFO logic to appear at external TEST_OUT. TEST_MUX[3–0] Internal Test Point Selected 0x00 rcv_ram_data[32] 0x01 atf_data_out[32] 0x02 itf_data_out[32] 0x03 atf_first_fct_rdy 0x04 itf_first_fct_rdy 0x05 grf_fct_rdy 0x06 force_itf_empty 0x07 grf_empty 0x08 grf_full 0x09 atf_empty 0x0a atf_thresh 0x0b itf_empty 0x0c itf_thresh 0x0d grf_6avail 0x0e grf_2avail 0x0f flush_apc
07–05	Not used	r	Return 0s when read.
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 is 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 is 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 is self-clearing.
01	FORCE_BIG_ENDIAN	r/w	When set to a 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 interpreted as a FIFO control token. FCT33_WR = 0 indicates that data pushed to bits 31–00 are normal data.

5.4.6 Asynchronous and Isochronous Transmit FIFO Threshold Control @A14

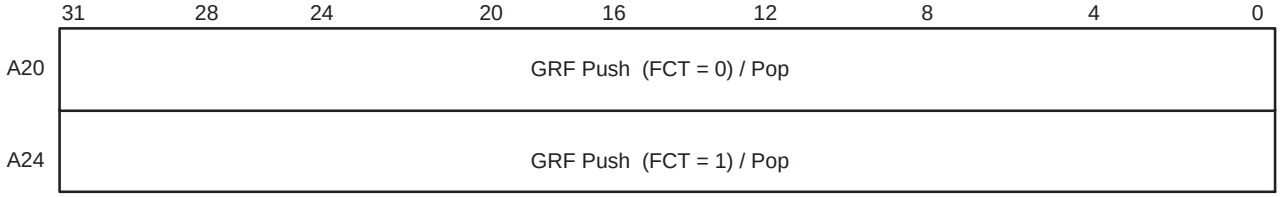
This register provides the application software with an interface for setting the transmit threshold of the ATF or ITF. When the number of data quadlets written into the ATF or ITF is greater than or equal to the number of quadlets specified by the corresponding threshold register, the 1394 link transmitter begins transmitting the packet from the FIFO whose threshold was triggered. This register is initialized to 0x0000FFFF on power-up reset. Since this value is larger than any possible FIFO size, no transfers will result. This register must therefore be set lower before normal transfers will occur. If it is set too low, then FIFO underruns or overruns will occur because the transmitter is triggered before sufficient data is in the FIFO. In general, the threshold register should be set to FIFO SIZE – CACHE LINE SIZE – 1.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–16	Reserved	r	Return 0s when read.
15–08	ATF_TRSHLD[7–0]	r/w	ATF transmit ready threshold in quadlets. Valid range = 0x00 to 0xFF
07–00	ITF_TRSHLD[7–0]	r/w	ITF transmit ready threshold in quadlets. Valid range = 0x00 to 0xFF

5.4.7 General Receive FIFO Data and Control Token Push-Pop @A20, A24

This port provides an interface for diagnostic software to write or read 33-bit data quadlets to or from the 33-bit wide general receive FIFO, via a PCI slave access.



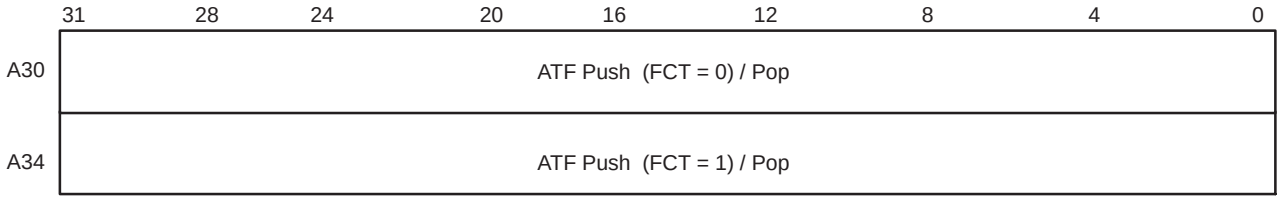
A write (PUSH) to address offset 0xA20 causes the 32-bit data quadlet to be written (pushed) to the 33-bit wide FIFO memory with MSB bit 32 set to a 0 and the 32-bit data quadlet written to bits 31–00. A write (PUSH) to address offset 0xA24 causes the 32-bit data quadlet to be written (pushed) to the 33-bit wide FIFO memory with MSB bit 32 set to a 1 and the 32-bit data quadlet written to bits 31–00.

A read (POP) from either offset 0xA20 or 0xA24 returns the data quadlet popped from bits 31–00 of the FIFO. The read also causes MSB bit 32 of the FIFO to be stored in the FIFO control token status register at offset 0xA0C bit 1. Software can then read the control token status register to determine the value that was last read (popped) from bit 32 of the FIFO memory.

FIFO bit 32 can also be observed on the TEST_OUT pin by programming the FIFO and PCI test muxes to select this bit. When the read (POP) is performed, the value of bit 32 will appear at the test mux output of the PCILynx chip during the active portion of the slave read cycle.

5.4.8 Asynchronous Transmit FIFO Data and Control Token Push-Pop Ports @A30, A34

This port provides an interface for diagnostic software to write or read 32-bit data quadlets to or from the 33-bit wide asynchronous transmit FIFO via a PCI slave access.



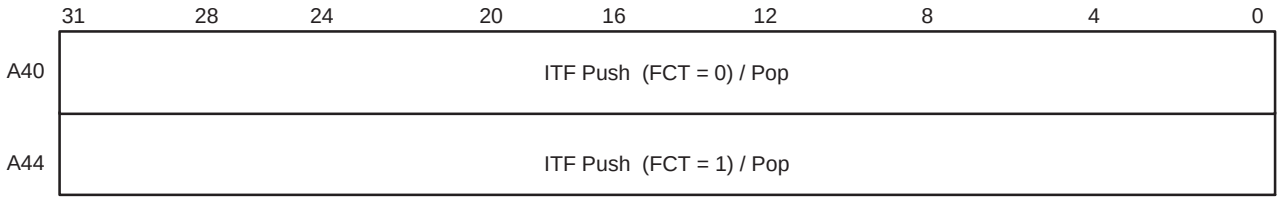
A write (PUSH) to address offset 0xA30 causes the 32-bit data quadlet to be written (pushed) to the 33-bit wide FIFO memory with MSB bit 32 set to a 0 and the 32-bit data quadlet written to bits 31–00. A write (PUSH) to address offset 0xA34 causes the 32-bit data quadlet to be written to the 33-bit wide FIFO memory with MSB bit 32 set to a 1 and the 32-bit data quadlet written to bits 31–00.

A read (POP) from either address offset will return the data quadlet stored in bits 31–00 of FIFO. The read also causes MSB bit 32 of the FIFO to be stored in the FIFO control token status register at offset 0xA0C bit 0. Software can then read the control token status register to determine the value that was last read (popped) from bit 32 of the FIFO memory.

FIFO bit 32 can also be observed on the TEST_OUT pin by programming the FIFO and PCI test muxes to select this bit. When the read (POP) is performed, the value of bit 32 will appear at the test mux output of the PCILynx chip during the active portion of the slave read cycle.

5.4.9 Isochronous Transmit FIFO Data and Control Token Push-Pop Ports @A40, A44

This port provides an interface for diagnostic software to write or read 32-bit data quadlets to or from the 33-bit wide isochronous transmit FIFO via a PCI slave access.



A write (PUSH) to address offset 0xA40 causes the 32-bit data quadlet to be written (pushed) to the 33-bit wide FIFO memory with MSB bit 32 set to a 0 and the 32-bit data quadlet written to bits 31–00. A write (PUSH) to address offset 0xA44 causes the 32-bit data quadlet to be written to the 33-bit wide FIFO memory with MSB bit 32 set to a 1 and the 32-bit data quadlet written to bits 31–00.

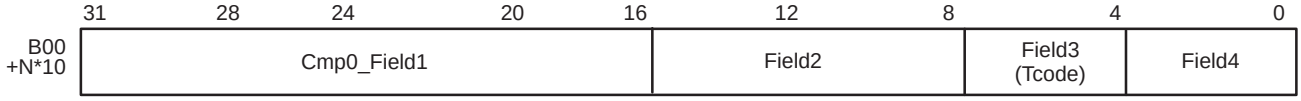
A read (POP) from either address offset will return the data quadlet stored in bits 31–00 of the FIFO. The read also causes MSB bit 32 of the FIFO to be stored in the FIFO control token status register at offset 0xA0C bit 0. Software can then read the control token status register to determine the previous value that was read from bit 32 of the FIFO memory location.

FIFO bit 32 can also be observed on the TEST_OUT pin by programming the FIFO and PCI test muxes to select this bit. When the read (POP) is performed, the value of bit 32 will appear at the test mux output of the PCILynx chip during the active portion of the slave read cycle.

5.5 1394 Link Layer Control and Status Register Definitions

5.5.1 DMA Channel 0 – 4 Word 0 Receive Packet Compare Value Register @B00, B10, B20, B30, B40

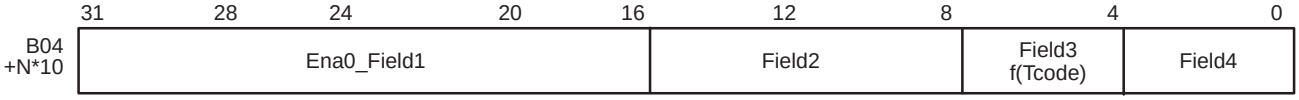
This register provides the interface for application software to program the compare-to-value which is used by the channel address comparator logic to match against the first word in the received packet. Bit 31 is the MSB.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–16	CMP0_FIELD1[15–0]	r/w	Specifies a 16-bit value to match against the high order 16 bits of the first quadlet of a received packet. For an asynchronous packet, this corresponds to the destination ID field; for an isochronous packet this corresponds to the data length field. This field is ignored if any bits of DEST_ID_SEL[4–0] are set.
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 asynchronous packet or the TAG/CHANNEL number field of an incoming isochronous packet.
07–04	CMP0_FIELD3[3–0]	r/w	Specifies an 4-bit value to match against the Tcode field on an incoming asynchronous or isochronous packet.
03–00	CMP0_FIELD4[3–0]	r/w	Specifies a 4-bit value to match against the PRIORITY field of an incoming asynchronous packet or the SYSTEM field of an incoming isochronous packet.

5.5.2 DMA Channel 0 – 4 Word 0 Receive Packet Compare Enable Register @B04, B14, B24, B34, B44

This register provides the interface for software to program the address comparator field enable register. This value is used to select the bit fields that are checked by the comparator logic when matching the value of the word 0 compare value registers to the first word of the incoming packet. Bit 31 is the MSB.



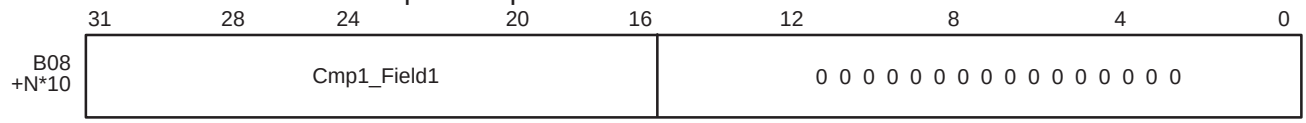
BIT NO.	BIT NAME	DIR	DESCRIPTION
31–16	CMP0_FIELD1_ ENABLE[15–0]	r/w	Specifies a 16-bit enable value for selecting the bit positions in the high order 16 bits of the first quadlet of a received packet to be compared to the value contained in the CMP0_FIELD1 register for a match. For an asynchronous packet, the field compared is the destination ID field; for an isochronous packet, the field compared is the data length field. A value of 1 in any bit position of the enable register, enables comparison for that bit position. A value of 0 disables comparison. This field is ignored if any bits of DEST_ID_SEL[4–0] are set. EXAMPLES: 0xFFXX of destination ID is selected to be compared against 0xFFXX in compare value register. 1st Quadlet of incoming packet FF50_0040 destination ID field = 0xFF50 CMP0_FIELD1 compare value register FF20 CMP0_FIELD1_ENABLE FF00 comparison is MATCH 0xXX50 of destination ID is selected to be compared against 0xXX20 of compare value register 1st Quadlet of incoming packet FF50_0040 destination ID field = 0xFF50 CMP0_FIELD1 compare value register FE20 CMP0_FIELD1_ENABLE register 00FF comparison is NOT MATCH
15–08	CMP0_FIELD2 _ENABLE[7–0]	r/w	Specifies an 8-bit enable register for selecting the bit positions in the transaction label/retry field of an incoming asynchronous packet that is compared against the corresponding bit positions of the CMP0_FIELD2 compare value register. If the incoming packet is isochronous then the TAG/CHANNEL number field in 1st quadlet of the packet is examined. A value of 1 in any bit position of the enable register, enables comparison for that bit position. A value of 0 disables comparison. EXAMPLES: 0x12 of on transaction/retry field is selected to be compared against 0x12 in compare value register 1st Quadlet of incoming packet FFC0_1240 transaction/retry field = 0x12 CMP0_FIELD2 compare value register 12 CMP0_FIELD2_ENABLE FF comparison is MATCH 0x1X of transaction/retry field is selected to be compared against 0x1X of compare value register 1st Quadlet of incoming packet FFC1_1240 transaction/retry field = 0x12 CMP0_FIELD2 compare value register 12 CMP0_FIELD2_ENABLE register F0 comparison is MATCH

BIT NO.	BIT NAME	DIR	DESCRIPTION
07–04	CMP0_FIELD3 _ENABLE[3–0]	r/w	Specifies a 4-bit code for programming the Tcode compare function mode which selects the GRF for receiving the incoming packet. 0000 = Reserved. Tcode compare is always true. 0001 = Reserved. Tcode compare is always true.* 0010 = Reserved. Tcode compare is always true.* 0011 = Compare Tcode of incoming packet for equality to CMP0_FIELD3[3–0]. 0100 = Compare Tcode of incoming packet for NOT equal to CMP0_FIELD3[3–0]. 0101 = Reserved. Tcode compare is always true.* 0110 = Compare Tcode of incoming packet for equality to CMP0_FIELD3[3–0] or that it is a 1394 asynchronous encoding except a cycle start Tcode.* 0111 = Reserved. Tcode compare match always true.* 1000 = Compare Tcode of incoming packet for not equal to CMP0_FIELD3[3–0] or that it is a 1394 async encoding except a cycle start Tcode.* 1001 = Compare Tcode of incoming packet for equal to 1394 isochronous encoding. 1010 = Compare Tcode of incoming packet for equal to any 1394 asynchronous Tcode encoding expect a cycle start Tcode. 1011 = 1111 Reserved. Tcode compare match always true.
03–00	CMP0_FIELD4 _ENABLE[3–0]	r/w	Specifies a 4-bit enable register for selecting the bit positions in the priority field of an incoming asynchronous packet that will be compared against the corresponding bit positions of the CMP0_FIELD4 compare value register. A value of 1 in any bit position of the enable register, enables comparison for that bit position. A value of 0 disables comparison. EXAMPLES: 0x1 of priority field is selected to be compared against 0x2 in compare value register 1st Quadlet of incoming packet FFC0_1241 priority field = 0x1 CMP0_FIELD4 compare value register 2 CMP0_FIELD4_ENABLE F comparison is NOT MATCH 0x2 of priority field is selected to be compared against 0x2 of compare value register 1st Quadlet of incoming packet FFC1_1242 priority field = 0x2 CMP0_FIELD4 compare value register 2 CMP0_FIELD4_ENABLE register F comparison is MATCH

* Definition different for PCILynx before Rev A

5.5.3 DMA Channel 0 – 4 Word 1 Receive Packet Compare Value Register @B08, B18, B28, B38, B48

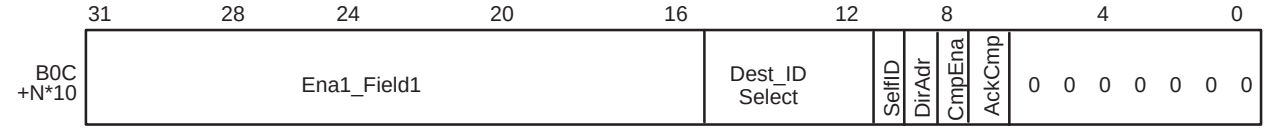
This register provides the interface for application software to program a compare-to-value which is used by the channel address comparator logic to match against the second word in an incoming packet. This register is initialized to 0x00000000 on power-up reset. Bit 31 is the MSB.



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 asynchronous packet
15–00	Reserved	r	Return 0s when read.

5.5.4 DMA Channel 0 – 4 Word 1 Receive Packet Compare Enable Register @B0C, B1C, B2C, B3C, B4C

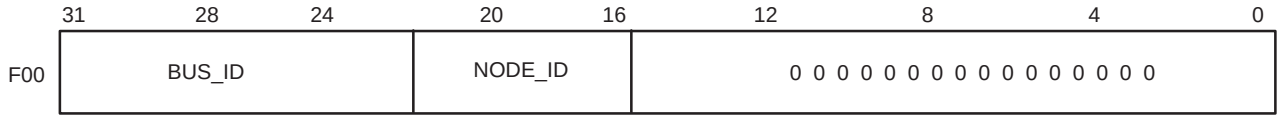
This register provides the interface for software to program the address comparator field select mask. This value is used to specify the bit fields that will be checked by the comparator logic when matching the value of the word 1 comparator register to the second word of the incoming packet. EN_CH_COMPARE and WRITE_REQ_ACK_SEL bits are initialized to 0s on power-up reset. Bit 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 asynchronous packet. This provides a first filter which all packets must get through if all bits of DEST_ID_SEL[4–0] are 0s. If any bit in DEST_ID_SEL[4–0] is set, then this value is ignored.
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 0x03FF
10	RCV_SELF_ID_EN	r/w	Enable reception of self-ID packets. Enable = 1, Disable = 0. When enabled, a packet will match a comparator if it is a self-ID or it matches any other programmed value.
09	EN_DIRECT_ADR	r/w	If this bit is set, then bits 15–0 of second quadlet received (quadlet 1) in a packet must be all 0s in order for the packet to match. Normally, this bit is set when the destination offset specified in quadlet 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 asynchronous packet. See Appendix E – *Program Control List (PCL) Examples.* (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 nonbroadcast write request packet will be ack'ed with an ack complete (0x0001). When set to 0, ack pending will be used (0x0010).
06–00	Reserved	r/w	Return 0s when read.

5.5.5 Bus Number and Node Number @F00

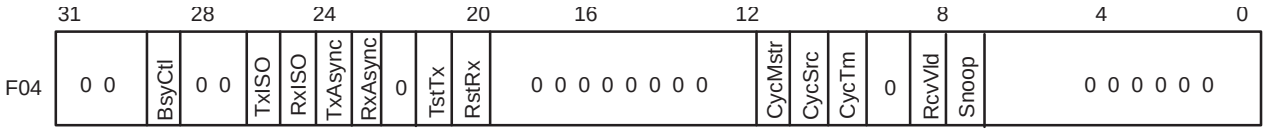
This register provides the interface for application software to program the 1394 bus and node identification numbers that are assigned to the PCILynx by the 1394 bus management layer. This register initializes to 0x00000000 on power-up reset. Bit 31 is the MSB.



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 0s when read.

5.5.6 1394 Link Layer Control @F04

This register provides 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 initializes to 0x00000000 on power-up reset. Bit 31 is the MSB.

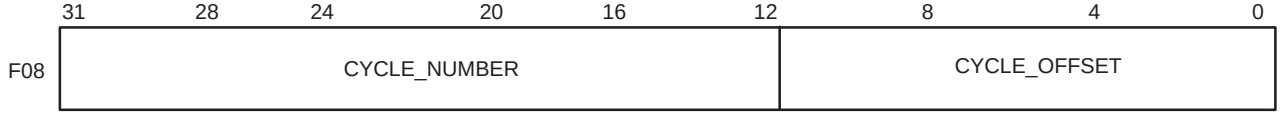


BIT NO.	BIT NAME	DIR	DESCRIPTION
31–30	Reserved	r	Return 0s when read.
29	BUSY_CNTRL	r/w	Controls what busy status the LLC will return on incoming packets that cannot be received. 0 = Use single phase busy protocol to busy incoming packets addressed to this node only when the GRF is unavailable. 1 = 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 0s when read.
26	TX_ISO_EN	r/w	Enable transmitter to send 1394 isochronous packets. enable = 1, disable = 0
25	RX_ISO_EN	r/w	Enable receiver to receive 1394 isochronous packets. enable = 1, disable = 0
24	TX_ASYNC_EN	r/w	Enable transmitter to send 1394 asynchronous packets. enable = 1, disable = 0 This bit is set to 0 when a bus reset event is detected by the link layer.
23	RX_ASYNC_EN	r/w	Enable receiver to receive 1394 asynchronous packets. enable = 1, disable = 0 This bit is set to 0 when a bus reset event is detected by the link layer.
22	Reserved	r	Returns 0 when read.
21	RSTTX	r/w	Reset 1394 transmitter. Reset = 1 causes synchronous reset of transmitter logic. This bit is self-clearing.
20	RSTRX	r/w	Reset 1394 receiver. Reset = 1 causes synchronous reset of receiver logic. This bit is self-clearing.
19–12	Reserved	r	Return 0s when read.
11	CYCMASTER	r/w	Enable PCILynx to be the cycle master. When set to 1, the LLC 1394 transmit logic sends a cycle start packet each time the cycle count field of the cycle timer register enable increments. Application software must not turn this bit on if the PCILynx is not attached to the ROOT PHY. This bit will automatically clear after a 1394 bus reset if the attached PHY is no longer ROOT.*
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 PCILynx 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. This bit must be set to 1 in order for packet transmissions to occur.
08	Not used	r/w	Returns 0 when 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. This bit must set to 1 in order for packet reception to occur.
06	SNOOP_ENABLE	r/w	When set to 1 the link receiver is placed in the SNOOP operating mode. In this mode, the address comparator logic is disabled. All 1394 packets (asynchronous, isochronous, and ack's) that are being transmitted on the bus by other nodes will be received and mapped to DMA channel 0.*
05–00	not used	r	Return 0s when read.

* PCILynx Rev A and higher

5.5.7 1394 Cycle Timer @F08

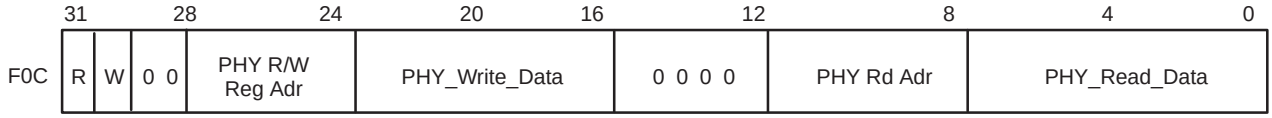
This register provides the interface for application software to program 1394 cycle timer counters with an initial value or read the current state of the counters. This register initializes to 0x00000000 on power-up reset. Bit 31 is the MSB.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–25	CYCLE_SECONDS[6–0]	r/w	Seconds portion of isochronous cycle number
24–12	CYCLE_COUNT[12–0]	r/w	Cycle count portion of isochronous cycle number. Increments every 125 µsec.
11–00	CYCLE_OFFSET[11–0]	r/w	24.576 MHz cycle timer counter – cycle offset rolls over every 125 µsec.

5.5.8 1394 Physical Layer Access F0C

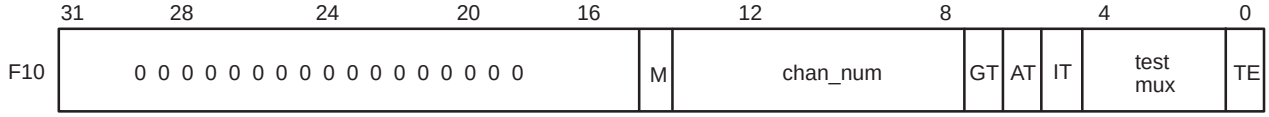
This register provides the interface for application software to access the control and status registers located in the physical layer chip. This register is initialized to 0x000000000 on power-up reset. Bit 31 is the MSB.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31	RDPHY	r/w	Read PHY register request. When set to 1, the LLC logic sends a read request to the PHY to return the value of the PHY register whose address is specified by bits 27–24 of this register. This bit is self-clearing.
30	WRPHY	r/w	Write PHY register request. When set to a 1, the LLC logic sends a write request to the PHY layer to write the 8-bit data specified in bits 23–16 of this register to the PHY address specified in bits 27–24. This bit is self-clearing.
29–28	not used	r	Return 0s when read.
27–24	PHY_REG_ADR[3–0]	r/w	Address of the PHY register to be written to or read
23–16	PHY_REG_DAT[7–0]	r/w	Data to be written to the PHY register address specified in bits 27–24 of this register.
15–12	not used	r	Return 0s when 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 it sends in response to a PHY register read request. This data was read from the address reported in bits 11–8 of this register. When reading a PHY register, software must verify that the returned address is the desired PHY register address before using this PHY read data.

5.5.9 1394 Diagnostic Test Control @F10

This register provides the interface for application software to perform diagnostic testing of the 1394 LLC functionality. This register initializes to 0x00000000 on power-up reset.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–15	Reserved	r	Return 0s when read.
14	CH_MATCH	r	This test point is the channel match indication generated by the link core address comparator logic.
13–08	DMA_CH_NO[5–0]	r	This test point is the 6 bit DMA channel number generated by the link core address comparator logic.
07	GRF_OFLOW_TST_STB	r/w	GRF overflow counter test increment. When a 1 is written to the bit, a one clock wide pulse is generated to the GRF overflow counter. This pulse causes the counter to increment by 1. This bit is 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 is generated to the ATF underflow counter. This pulse causes the counter to increment by 1. This bit is 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 is generated to the ITF underflow counter. This pulse causes the counter to increment by 1. This bit is self-clearing.
04–01	TESTMUXSEL[3–0]	r/w	Select internal test point for observation at the external TEST_OUT pin TESTMUXSEL[3–0] — Link core signal selected 0x0000 — rxDataRdy 0x0001 — rxDataErr 0x0010 — rxDataEnd 0x0011 — busReqIso 0x0100 — busReqPri 0x0101 — busReqFair 0x0110 — busGrant 0x0111 — busBusy 0x1000 — txdDataEn 0x1000 — txMore 0x1010 — sendAck 0x1011 — ackSent 0x1100 — readyAck 0x1101 — ch_match 0x1110 — arb_gap 0x1111 — sub_action_gap
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 enables the specific diagnostic test modes defined in this register definition.

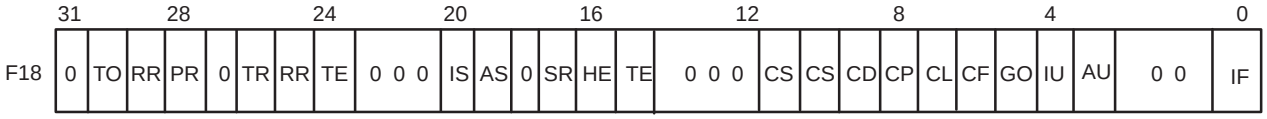
5.5.10 1394 Link Layer Interrupt Status Register @F14

This register provides the interface for application software to determine the interrupt that is caused by a 1394 link event. This register is set to 0x00000000 on power-up reset. An interrupt bit is asserted when it is set to a logic 1. The interrupt status bits defined in the following table is cleared by writing a 1 to a selected interrupt bit. The bits defined in this register will not set if the corresponding interrupt enable bit is set to 0. Bit 31 is the MSB.

BIT NO.	BIT NAME	DIR	DESCRIPTION
31	LINK_INT	r/w	Link logic interrupt. This signal is the logical OR of all link interrupt sources.
30	PHY_TIME_OUT	r/w	The PHY has stayed in a particular state for too 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	Returns 0 when read.
26	TX_RDY	r/w	The transmitter has completed sending a packet.
25	RX_DATA_RDY	r/w	The receiver has received a packet.
24–21	Reserved	r	Return 0s when read.
20	IT_STUCK	r/w	Transmitter stuck on isochronous transfer from ITF.
19	AT_STUCK	r/w	Transmitter stuck on asynchronous transfer from ATF.
18	Reserved	r	Returns 0 when read.
17	SNTRJ	r/w	The receiver was forced 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 incoming 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 0s when 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 isochronous cycle is over.
08	CYC_PEND	r/w	Cycle pending is asserted when cycle timer offset is set to 0 (rolled over or reset) and stays asserted until the isochronous cycle has ended.
07	CYC_LOST	r/w	Cycle timer has rolled over twice 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	The GRF_OVER_FLOW counter at offset 0xF24 reached 0xFF. Counter is incremented with each GRF overflow detected during packet reception. Writing a 1 to this bit also clears the GRF_OVER_FLOW counter at offset 0xF24 bits 7–0.
04	ITF_UNDER_FLOW	r/w	The ITF_UNDER_FLOW counter at offset 0xF24 reached 0xFF. Counter is incremented with each ITF under flow detected during isochronous packet transmission. Writing a 1 to this bit also clears the ITF_UNDER_FLOW counter at offset 0xF24 bits 23–16.
03	ATF_UNDER_FLOW	r/w	The ITF_UNDER_FLOW counter at offset 0xF24 reached 0xFF. Counter is incremented with each ATF underflow detected during asynchronous packet transmission. Writing a 1 to this bit also clears the ITF_UNDER_FLOW counter at offset 0xF24 bits 23–16.
02–01	Reserved	r/w	Return 0s when read.
00	IARB_FAILED	r/w	Arbitration to send an isochronous packet has failed.

5.5.11 1394 Link Layer Interrupt Enable Register @F18

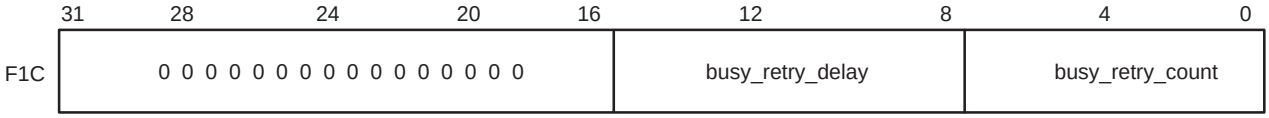
This register provides the interface for application software to enable the interrupt specified in the 1394 link layer interrupt status register. This register is set to 0x00000000 on power-up reset. Bit 31 is the MSB. Setting an enable bit to a logic 1 enables the interrupt. Setting the enable bit to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



BIT NO.	BIT NAME	DIR	DESCRIPTION
31	Reserved	r	Returns 0 when 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	Returns 0 when 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 0s when read.
20	IT_STUCK_EN	r/w	Enable transmitter stuck on isochronous interrupt.
19	AT_STUCK_EN	r/w	Enable transmitter stuck on isochronous interrupt.
18	Reserved	r	Returns 0 when 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 0s when read.
11	CYC_SEC_EN	r/w	Enable cycle timer seconds interrupt.
10	CYC_STRT_EN	r/w	Enable 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 0s when read.
00	IARB_FAILED_EN	r/w	Enable isochronous arb failed interrupt

5.5.12 1394 Busy Retry Control Register @F1C

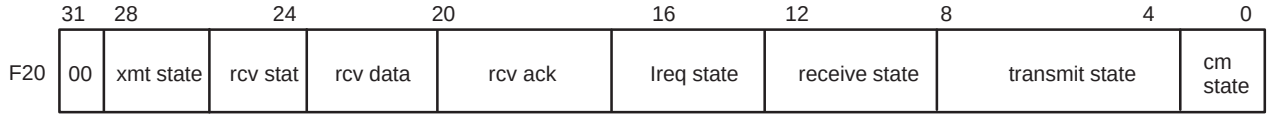
This register provides the interface for application software to set the number of times that the 1394 transmitter is to retry an asynchronous packet that was acknowledged with a busy or error condition. This register also provides the means to program the time interval to delay between successive retries. Bit 31 is the MSB. This register is cleared to all 0s on power-up reset.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–16	Reserved	r	Return 0s when read.
15–08	BUSY_RETRY_DLY[7–0]	r/w	A number between 0 and 255 that specifies the time that the 1394 transmitter must delay between successive retries. This time is equal to BUSY_RETRY_DLY[7–0] times ISO_INTERVAL (125 µsec). Any nonzero value in this register requires that CYCTIMEREN be set otherwise a retried cycle will hang.
07–00	BUSY_RETRY_CNT[7–0]	r/w	A number between 0 and 255 that specifies the maximum number of times to re-transmit a packet, when the destination node continues to return busy acknowledge status. The 1394 transmitter notifies the active DMA channel when the maximum number of transmit retries have been attempted without a successful transmission occurring.

5.5.13 Link Layer Controller State Machine Vector Monitor Port @F20

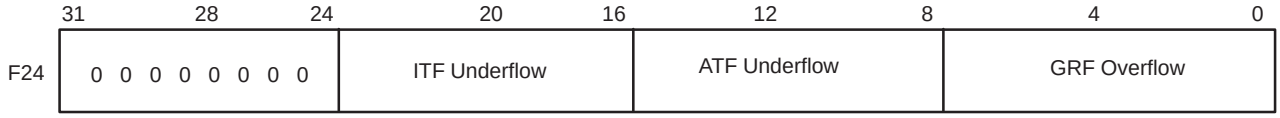
This register provides application software with an I/O port to read the value of the state vector for each state machine in the link layer control logic. This register is read-only.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–30	Not Used	r	Return 0s when read.
29–26	TRANSMIT_IFC_STATE[3–0]	r	TransmitIfc state machine vector
25–24	RXSTATUS_IFC_STATE[1–0]	r	RxdStatIfc state machine vector
23–21	RXDDATA_IFC_STATE[2–0]	r	RxdDataIfc state machine vector
20–18	RCV_ACK_STATE[2–0]	r	RcvAck state machine vector
17–14	LREQ_STATE[3–0]	r	Request state machine vector
13–09	RECEIVE_STATE[3–0]	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

5.5.14 Link Layer FIFO Under Flow – Over Flow Counters @F24

These counters 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 0s on power-up reset, and are also cleared by writing a 1 to the appropriate bit GRF_OVER_FLOW, ITF_UNDER_FLOW or ATF_UNDER_FLOW bit in the link layer interrupt status register. Bit 31 is the MSB.



BIT NO.	BIT NAME	DIR	DESCRIPTION
31–24	Not used	r	Return 0s when read.
23–16	ITF_UNDER_FLOW[7–0]	r/w	Increments by 1 on each ITF underflow detected by the 1394 transmitter or when a 1 is written to the ITF_UFLOW_TST_STB bit in the 1394 diagnostic test control @F10. When 0xFF is reached, the ITF_UNDER_FLOW interrupt bit of the 1394 link layer interrupt status register @F14 is set and counting stops and holds at 0xFF. Subsequent underflows will not increment the counter. The counter is enabled when software loads it with any value that is less than 0xFF. Counting will proceed from that value until 0xFF is reached. This counter is cleared when the ITF_UNDER_FLOW interrupt bit is cleared.
15–08	ATF_UNDER_FLOW[7–0]	r/w	Increments by 1 on each ATF underflow detected by the 1394 transmitter or when a 1 is written to the ATF_UFLOW_TST_STB bit in the1394 diagnostic test control @F10. When 0xFF is reached, the ATF_UNDER_FLOW interrupt bit of the 1394 link layer interrupt status register @F14 is set and counting stops and holds at 0xFF. Subsequent underflows will not increment the counter. The counter is enabled when software loads it with any value that is less than 0xFF. Counting will proceed from that value until 0xFF is reached. This counter is cleared when the ATF_UNDER_FLOW interrupt bit is cleared.
07–00	GRF_OVER_FLOW[7–0]	r/w	Increments by 1 on each GRF overflow detected by the 1394 receiver or when a 1 is written to the GRF_OFLOW_TST_STB bit in the 1394 diagnostic test control @F10. When 0xFF is reached, the GRF_OVER_FLOW interrupt bit of the 1394 link layer interrupt status register @F14 is set and counting stops and holds at 0xFF. Subsequent overflows will not increment the counter. The counter is enabled when software loads it with any value that is less than 0xFF. Counting will proceed from that value until 0xFF is reached. This counter is cleared when the GRF_OVER_FLOW interrupt bit is cleared.

Appendix A Signal to Package Assignments

GROUND PINS = 22; 3.3 V VCC PINS = 21; 5 V REFERENCE PINS = 8;
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	5.0V VCC	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_rdyz	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

Table A–1. PCILynx I/O Signal Function Table

Signal Name	Dir	Note	Output Drive (mA)	Functional Description
GND	I	1		Ground
3.3V VCC	I	1		3.3 Volt power
5.0V VCC	I	1		5 Volt tolerance input
pci_clk	I	1		PCI system clock. 0–33 MHz
pci_ad[31:0]	I/O	1		PCI multiplexed address/data bus signals
pci_cbez[3:0]	I/O	1		PCI multiplexed command/byte enable signals
pci_par	I/O	1		PCI parity signal. Parity is even across pci_ad [31:0] and pci_cbez[3:0] signals
pci_framez	I/O	1		PCI frame signal
pci_irdyz	I/O	1		PCI initiator ready signal
pci_trdyz	I/O	1		PCI target ready signal
pci_devselz	I/O	1		PCI device select
pci_stopz	I/O	1		PCI stop
pci_idsel	I	1		PCI initialization device select
pci_perrz	I/O	1		PCI data parity error
pci_serrz	OD	1		PCI system error. This is an open drain signal.
pci_reqz	O	1		PCI master bus request to PCI bus arbiter
pci_gntz	I	1		PCI bus grant from PCI bus arbiter
pci_resetz	I	1		PCI system reset
pci_intaz	OD	11		PCI system interrupt A. This is an open drain signal
seeprom_data	I/O	2, 3	4	External serial EEPROM read-write data line
seeprom_clk	I/O	2, 4	4	External serial EEPROM data clock
aux_clk	O	5	8	Auxiliary port clock out (output at frequency of PCI Clock)
aux_rstz	O	5	4	Auxiliary port reset out
aux_intz	I	6		Auxiliary port interrupt in
gpio_data[3:0]	I/O	3	4	Auxiliary port general purpose programmable i/o signals
aux_adr[15:0]	O	5	4	Auxiliary port address lines out to external logic
aux_data[15:0]	I/O	3	4	Auxiliary port bidirectional data bus to external logic
aux_oez	O	5	4	Auxiliary port output enable to enable external logic data on to the aux_data bus
aux_wez[1:0]	O	5	4	Auxiliary port write strobes to external logic
aux_rdyz	I	6		Auxiliary port ready indication from external logic
aux_csz	O	5	4	Auxiliary port chip select to external logic
rom_csz	O	5	4	External ROM chip select
ram_csz	O	5	4	External RAM chip select
phy_ctl[0:1]	I/O	7	12	Phy-link bidirectional control lines
phy_data[0:7]	I/O	3	12	Phy-link bidirectional data lines
phy_clk50	I	1		50 MHz system clock from PHY chip.
phy_lreq	O	1	12	Phy-link request signal generated by the PCILynx chip
link_isoz	I	8		Phy-link isolation barrier mode
link_cyclein	I	10		Optional external 8 kHz clock for use as the cycle clock.
link_cycleout	O	5	4	Cycle timer 8 kHz cycle clock out
zv_ext_clk	I	6		Zoom port external clock input

Table A–1. PCILynx I/O Signal Function Table (Continued)

zv_vsync	O	3	4	Zoom port vertical sync output
zv_hsync	O	3	4	Zoom port horizontal sync output
zv_data_valid	O	3, 9	4	Zoom port data valid signal
test_out	O	5	4	Test mux out. Internal test point selected by the test multiplexer for observation.
test_enable	I	10		Test enable. Enables factory test features.
autoboot	I	10		Autoboot. Selects autoboot mode.
zv_pix_clk	O	3	4	Zoom port pixel clock.

NOTES: 1. Required connection
2. Bidirectional with open-drain style output. Connect through resistor to same V_{CC} as SEEPROM uses.
3. Connect to 3.3 V V_{CC} or GND through resistor if this pin not used in system.
4. Connect to GND through resistor for SEEPROM not present detection (see SEEPROM control register @044).
5. May be left unconnected if not used in system.
6. Connect to 3.3 V V_{CC} or GND directly or through a resistor if not used in system.
7. phy_ctl pins should be connected to physical layer device and have a resistor to GND.
8. Isolation buffers are not implemented. This pin must be connected to 3.3 V V_{CC}. Shown as 3.3 V V_{CC} on datasheet.
9. If Zoom port is used without zv_data_valid pin, must use gated clock mode.
10. Connect to GND directly or through a resistor if not used in system.
11. From a hardware standpoint PCILynx is capable of sharing an interrupt line, however this may lead to reduced performance due to multiple interrupt service routines being called for each interrupt.

I/O Characteristics

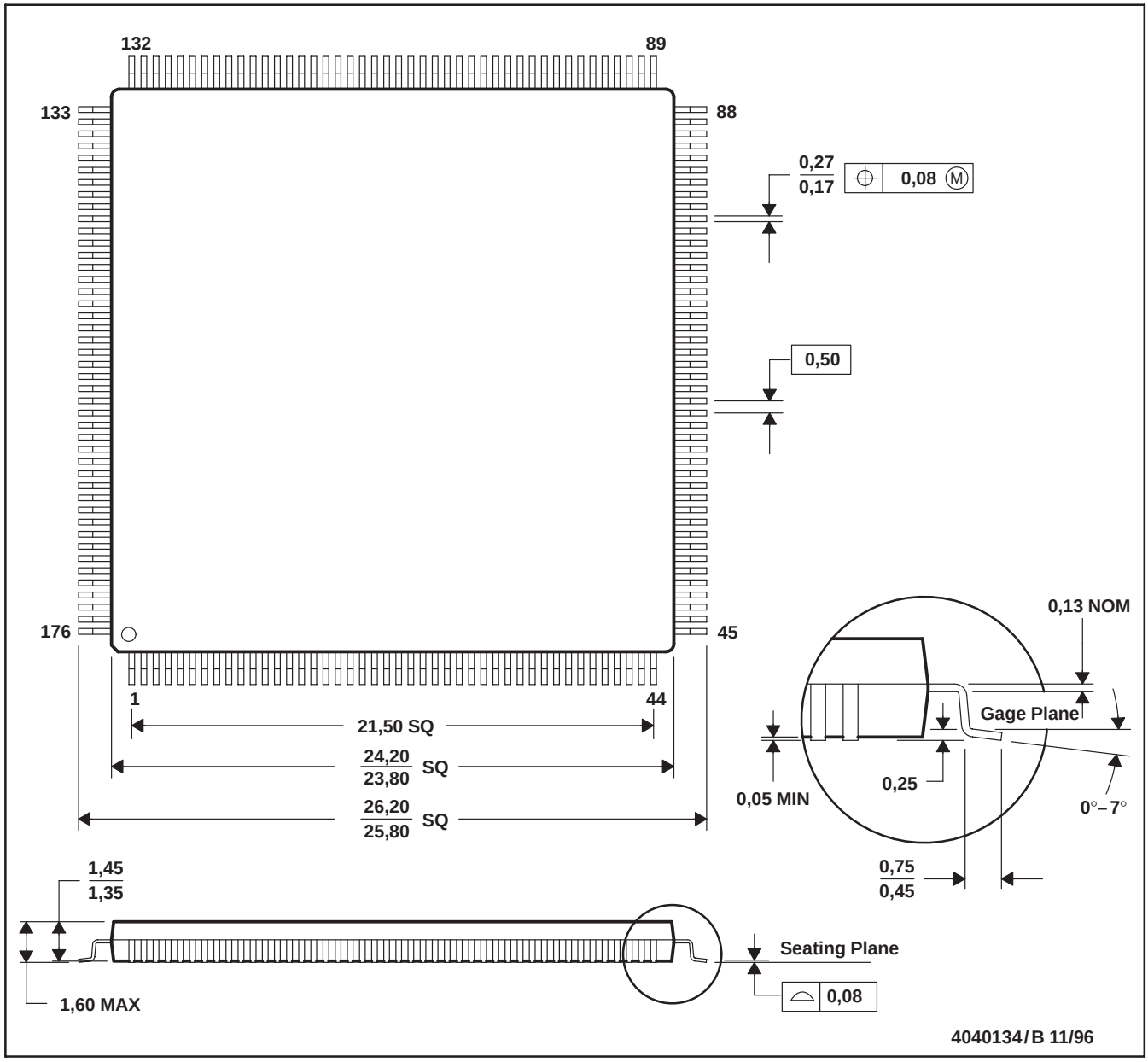
All signal pins (except for test_out) contain an input buffer for test purposes even if the signal is functionally an output. The guidelines given above for tying off unused connections must be followed to prevent high through-current in the input buffer.

PCILynx incorporates 5-V tolerant inputs on all signal pins. Due to the unique characteristics of these buffers a small amount of current may flow from 3.3-V V_{CC} through the input pin under certain conditions. When an I/O is terminated through a resistor to GND, the resistance should be 1.8 kΩ or less. Therefore the recommended value for all resistors to GND is 1.8 kΩ. Resistors to 3.3-V V_{CC} have no such restriction. Resistors to the 5-V V_{CC} are not recommended except for the SEEPROM pins.

The 5-V tolerant inputs are tolerant to voltages over 3.6 V only when the 3.3 V V_{CC} is on. This has important implications for power supply sequencing. Please see Appendix G for more details.

Appendix B ASIC Package Outline Dimension Drawing

PGF (S-PQFP-G176) *PLASTIC QUAD FLATPACK*



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS-026

Figure B–1. 176 Pin Plastic Quad Flat Pack (S-PQFP-G176)

Appendix C FIFO Packet Organization Formats

Table C–1. 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 LOW				
0	DESTINATION OFFSET HI					
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 D.
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 D.

Table C–2. 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 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 D.
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 D.

Table C–3. 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					

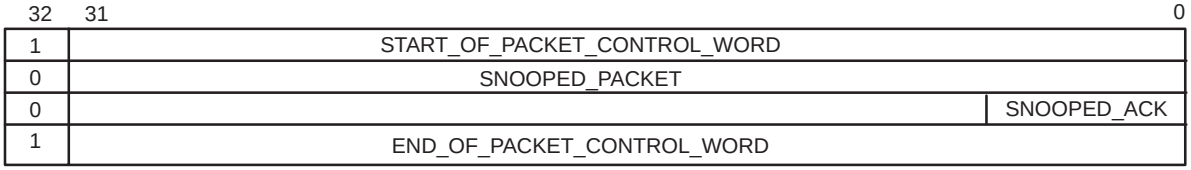
Field Name	Bit Positions	Description
START_OF_PACKET CONTROL_WORD	31 – 0	FIFO start control word. Marks the start of the packet in the FIFO. Contains control information that is set up by the DMA channel for use by the 1394 transmitter logic. The bit field definitions for this control word are specified in Appendix D.
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 for this packet. 00=nre, 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 D.

Table C–4. 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 D.
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 D.

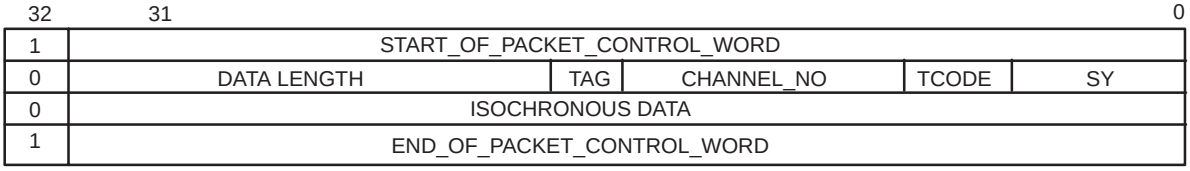
Table C–5. General Receive FIFO Snoop Mode Packet Format*



Field Name	Bit Positions	Description
START_OF_PACKET CONTROL_WORD	31 – 0	FIFO start control word. Marks the start of a snooped packet in the FIFO. Contains control information that will be used by DMA channel 0 for controlling the transfer of the snooped packets from the FIFO to PCI host memory. The details of this control word are specified in Appendix D.
SNOOPED_PACKET	31 – 0	N number of quadlets that comprise the packet that was snooped. The SNOOPED_PACKET will include any 1394 header CRC or payload CRC quadlets. These CRC quadlets are not received in a normal receive operation.
SNOOPED_ACK	3–0	The 4-bit ack status that was snooped. If the link receiver does not detect a ack for the snooped packet, the SNOOPED_ACK value will be set to 0000.
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 used by DMA channel 0 for transferring the snooped packet from the FIFO to host memory. The bit field definitions for this control word are specified in Appendix D.

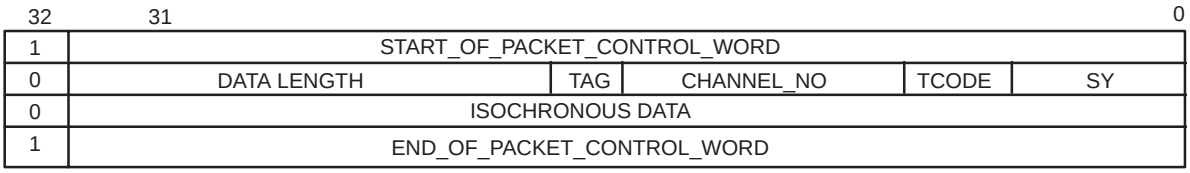
*PCILynx Pev A and Higher

Table C–6. Isochronous Transmit FIFO Packet Format



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 D.
DATA LENGTH	31 – 16	Indicates the number of bytes in the ISO packet
TAG	15 – 14	Tag field
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 D.

Table C–7. Isochronous Receive FIFO Packet Format



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 D.
DATA LENGTH	31 – 16	Indicates the number of bytes in the ISO packet
TAG	15 – 14	Tag field
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 D.

Appendix D FIFO Control Word and Transmit ACK Formats

Table D–1. General Receive FIFO Isochronous Packet Control Token Format Definition

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

Bit Field	Function Description
FCT	FIFO control token. FCT = 1 indicates that FIFO data bits 31 – 0 of the quadlet are to be interpreted as a FIFO control token. The bit fields defined below for this control token shall only be valid when FCT = 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, 10 = 400 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. When the PCILynx 1394 receiver is running in SNOOP MODE, the DMA channel number will be set to 000000. This field is 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

Table D–2. General Receive FIFO Asynchronous Packet Control Token Format Definition

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

Bit Field	Function Description
FCT	FIFO control token. FCT = 1 indicates that bits 31 – 0 of the quadlet are to be interpreted as a FIFO control token. The bit fields defined below for this control token shall only be valid when FCT = 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. When the PCILynx 1394 receiver is running in SNOOP MODE, the DMA channel number will be set to 000000. 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

Table D–3. Isochronous Transmit FIFO Control Word Format

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

Bit Field	Function Description
FCT	FIFO control token. FCT = 1 indicates that bits 31 – 0 of the quadlet are to be interpreted as a FIFO control token. The bit fields defined below for this control token shall only be valid when FCT = 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 – 100 mbps SPD_CODE = 01 – 200 mbps 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.

Table D–4. Isochronous Transmit FIFO Control Word Format

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

Bit Field	Function Description
FCT	FIFO control token. FCT = 1 indicates that bits 31 – 0 of the quadlet are to be interpreted as a FIFO control token. The bit fields defined below for this control token shall only be valid when FCT = 1.
PKTBNDRY	Packet delimiter. PKTBNDRY = 00 indicates start of packet PKTBNDRY = 10 indicates end of packet
SPD_CODE	Transmit speed code. SPD_CODE = 00 – 100 mbps SPD_CODE = 01 – 200 mbps 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.

Table D–5. 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.
0	0	1	1	1	Acknowledge packet error – An a parity error or a truncation occurred during the reception of the 8-bit transmit acknowledge packet.
1	1	1	0	1	Improper packet format – Packet was not transmitted because of a malformed header.

Appendix E Program Control List (PCL) Examples

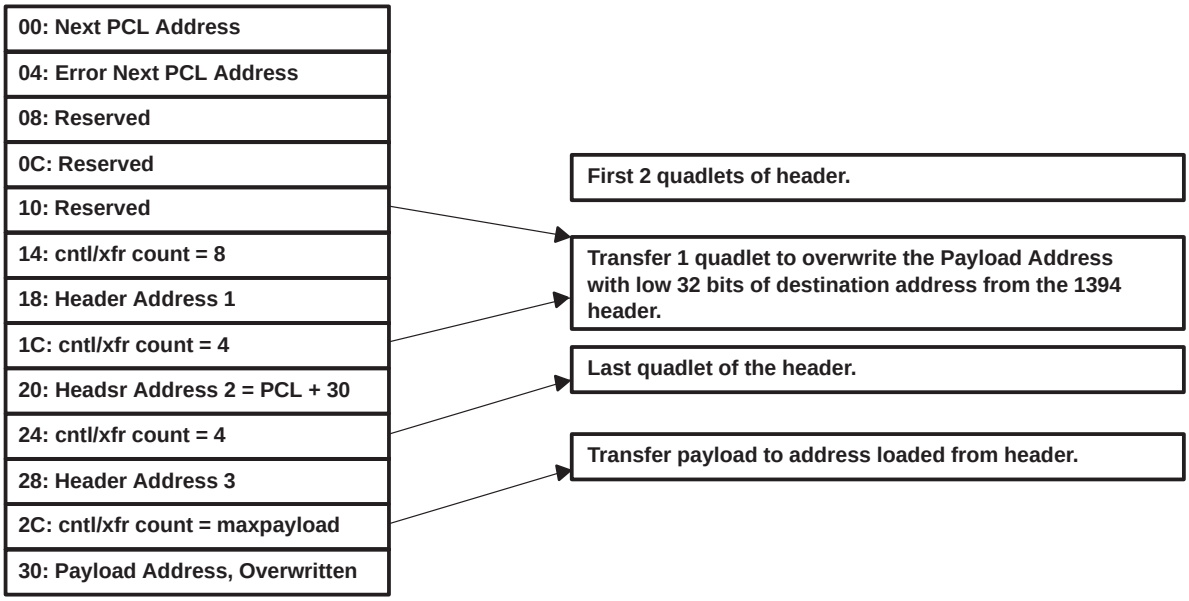
E.1 Transfer AT ADDRESS Program

The PCILynx PCLs 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 so as to modify the data buffer address for the payload data.

For example: For a write request data block:

Allocate a DMA channel for this purpose (i.e., dedicated to processing write request data block transactions only), and setup the receive comparators to allow only accesses from a trusted node(s), only write request data block Tcodes, and only requests with the destination address high equal to 0. Set PCILynx to return ACK_COMPLETE for accesses to this DMA channel.

Example PCL:



E.2 PHY Configuration Packets

A PHY Configuration (PHY CFG) packet (or any other PHY packet or any other packet) may be transmitted by the PCILynx by using the UNFXMT command. The two quadlets described in the 1394 Standard can be transmitted as the 8-byte payload of an ASYNC packet. This will always result in an ACK_TIMEOUT completion status since PHY packets are not ACK'ed on the 1394 bus, and the async transmitter used to transmit this format is expecting a returned ACK. (If there were an ACK returned, then it would be returned in the completion status with no error).

Table E–1. PHY_CFG Packet (Big Endian Format)

PCI Address	PCL Description	Value	Comment
0001FC30 00	NEXT_PCL	0001B461	NOT_VALID=1
0001FC34 04	ERROR_PCL	0001B461	
0001FC38 08	SOFTWARE	00000000	
0001FC3C 0C	STATUS	00000000	CH=0 SPD=0 ACK=0=ACK_RSVD_0 XFR=000
0001FC40 10	RCV_COUNT	00000000	RCV_AND_UPDATE command only
0001FC44 14	NXT_BUF_ADR	00000000	RCV_AND_UPDATE command only
0001FC48 18	PAIR 0 CTRL	0C070008	CMD=C=UNFXMT, LASTBUF, Wait for status, big endian, 8 Bytes
0001FC4C 1C	PAIR 0 ADDR	@0001FC50	
@0001FC50 20	QUADLET DATA 0	01F00000	R=1, T=1, GAPCNT=30 (big endian format)
0001FC54 24	QUADLET DATA 1	FE0FFFFF	Inverted QUAD0 (big endian format)

Table E–2. PHY_CFG Packet (Little Endian Format)

PCI Address	PCL Description	Value	Comment
0001FC30 00	NEXT_PCL	0001B461	NOT_VALID=1
0001FC34 04	ERROR_PCL	0001B461	
0001FC38 08	SOFTWARE	00000000	
0001FC3C 0C	STATUS	00000000	CH=0 SPD=0 ACK=0=ACK_RSVD_0 XFR=000
0001FC40 10	RCV_COUNT	00000000	RCV_AND_UPDATE command only
0001FC44 14	NXT_BUF_ADR	00000000	RCV_AND_UPDATE command only
0001FC48 18	PAIR 0 CTRL	0C060008	CMD=C=UNFXMT, LASTBUF, Wait for status, 8 Bytes
0001FC4C 1C	PAIR 0 ADDR	@0001FC50	
@0001FC50 20	QUADLET DATA 0	0000F001	R=1, T=1, GAPCNT=30 (little endian format)
0001FC54 24	QUADLET DATA 1	FFFF0FFE	Inverted QUAD0 (little endian format)

Appendix F Serial EEPROM Data

The PCILynx 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 PCILynx will be terminated with a retry disconnect. This ensures that the system software will always read the values loaded from serial EEPROM whenever the PCILynx is first accessed.

If the serial EEPROM data is not loaded, the default values shown will persist after PCI reset. A checksum error will only set the EEPCHKERR bit of the serial EEPROM control register @044; it will not prevent the serial EEPROM data from loading.

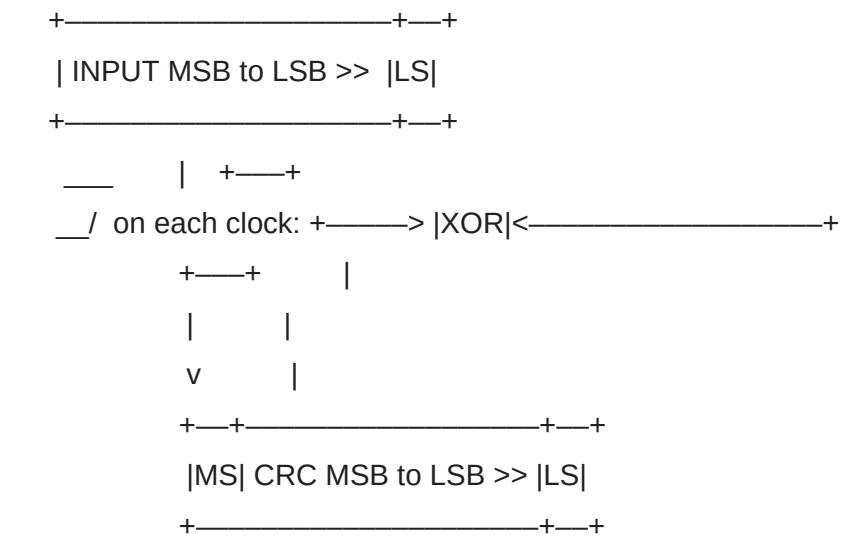
The first 8 bytes of the serial EEPROM address space are reserved for use by the PCILynx after power reset. These bytes are loaded into internal registers by the PCILynx. This is described in the following table. The second 8 bytes are reserved for future use by the hardware. All remaining bytes in the 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 PCILynx.

Some may desire to eliminate the serial EEPROM to reduce system cost. While this is possible from a hardware standpoint, careful consideration must be given to applicable system specifications. Some of the register locations can only be loaded from the serial EEPROM and therefore could not be changed from their default value if the serial EEPROM is eliminated. Also consider the requirements of any software to be used with PCILynx.

Table F–1. Serial EEPROM Address Map

Byte Adr	Default	Byte Description
00	x02	PCI max_lat (Configuration Reg 3F)
01	x01	PCI min_gnt (Configuration Reg 3E)
02	xE0	Local bus control register – ROM control (Configuration Reg B0)
03	x00	PCI subsystem vendor ID (ls byte) (Configuration Reg 2C)
04	x00	PCI subsystem vendor ID (ms byte) (Configuration Reg 2D)
05	x00	PCI subsystem ID (ls byte) (Configuration Reg 2E)
06	x00	PCI subsystem ID (ms byte) (Configuration Reg 2F)
07		Checksum (bytes 0–6)
08		Reserved for future hardware use
...		.
0F		Reserved for future hardware use
10		User defined
...		.
FF		User defined

The algorithm that should be used to generate the CRC code placed in the serial EEPROM is included here.



CRC SEED = 10101010 (hex 0xAA)

It is assumed that data in the serial EEPROM will be shifted LSB to MSB consistent with other serial communication streams.

Below is a c routine to calculate the serial EEPROM crc over the first 7 bytes of the EEPROM. The return value is written to the 8th byte of the EEPROM. The 'msb' arg passed is a copy of the bytes read from the EEPROM and stored in an array.

```
unsigned char
xor_crc(unsigned char *msb, int bytes)
{
  unsigned char crc=0xaa; // seed
  int i;
  for(i=0; i<bytes; i++)
    crc^=msb[i];
  return(crc);
}
```

Appendix G Power Supply Sequencing

Turning power supplies on and off within a mixed 5-V/3.3-V system is an important consideration. Observe a few basic rules to avoid damaging PCI-Lynx devices. Please check with the manufacturers of all components used in the 3.3-V to 5-V interface to ensure that no unique device characteristics exist that would lead to rules more restrictive.

If the 3.3-V supply is turned on before turning on the 5-V supply, PCI-Lynx output buffers in a logic 1 state can supply large amounts of current through their clamp diodes to the 5-V supply pin. This can lead to excessive power dissipation and a violation of current density limits. However, if the 5-V supply is turned on before the 3.3-V supply, the maximum drain-to-gate voltage of the n-channel transistors in the 5-V tolerant buffers exceeds the recommended value, and the effects of channel hot carriers can be accelerated.

When turning on the power supply, all 3.3-V and 5-V supplies should start ramping from 0 V and reach 95 percent of their end-point values within a 25-ms time window. All bus contention between the PCI-Lynx and external devices is eliminated by the end of the 25-ms time window. The preferred order of supply ramping is to ramp the 3.3-V supply followed by the 5-V supply. This order is not mandatory, but it allows a larger cumulative number of power supply events than the reverse order.

When turning off the power supply, all 3.3-V and 5-V supplies should start ramping from steady state values and reach 5 percent of these values within a 25-ms time window. All bus contention between the PCI-Lynx and external devices is eliminated by the end of the 25-ms time window. The preferred order of supply ramping is to ramp down the 5-V supply followed by the 3.3-V supply. This order is not mandatory, but it allows a larger cumulative number of power supply off events than the reverse order.

A cumulative total of 250 seconds of power supply turn-on or turn-off events is allowed during the operating lifetime of the PCI-Lynx under worst-case conditions (where the 5-V supply is ramped up before the 3.3-V supply, and the 3.3-V supply is ramped down before the 5-V supply). If the maximum time window of 25 ms is used, a total of 10000 power supply on/off events can occur as long as the 25-ms time window is observed.

An additional precaution must be observed when the PCI-Lynx is attached to a 5-V IEEE 1394 physical layer device which is powered from the 1394 cable. In that case, it is possible for the physical layer device to have power while the PCI-Lynx does not. It is essential that the physical layer device must not supply a high on any pin which connects to the PCI-Lynx while the PCI-Lynx power is off. This is normally achieved through the use of the link power status pin on the physical layer device.

CAUTION:

If these precautions and guidelines are not followed, the PCI-Lynx device may experience possible failures related to overheating, accumulation of channel hot carriers, and/or metal migration due to excessive current density.

Appendix H Device Changes

Revision A device changes

The following paragraphs describe the Revision A changes to the TSB12LV21. This device is now known as the TSB12LV21A.

DMA Swap & Compare command added

The Rev A PCILynx has an added DMA auxiliary command called swap & compare which acts like the normal compare but first swaps the 16-bit halves of the temp register. See Table 2, *AUXILIARY Command Packet Control List Format* and section 4.2.2, *DMA Logic*.

Interrupt bit definition changed

The Int (Interrupt) bit definition in the PCL control and byte count quadlet changed slightly in that it is used in the first transfer scatter pair of a PCL. Subsequent scatter entries' Int bit are ignored. See the Int bit description in the transfer command data buffer 0 control register on page 27.

Lower Bound register redefined

The lower bound register was redefined. See section 5.3.10, *Global Register @908*.

DMA performance enhancements

DMA performance was enhanced by increasing the number of context switching opportunities in the DMA state machine.

PCI revision ID changed

Revision ID changed to 0x02. See section 5.2.3, *Class Code – Revision ID @008*.

ZOOM port ZV_PIX_CLK

Polarity control for the ZOOM Port ZV_PIX_CLK was added to the local bus control register. See section 5.2.16, *Local Bus Control Register @0B0 {ROM, RAM, AUX, and ZV registers}*. Bit 19 is the "invert ZV clock" bit.

Also, when autoboot is active, PCI reset enables and selects ZV_PIX_CLK as PHY_CLK50/2 (25 MHz). This should allow a standalone application to use the ZV_PIX_CLK for a PCI clock.

ZOOM port Vertical sync output

The prior PCILynx vertical sync output was not properly recognized by some video controllers and GPIO DATA3 was used instead for the vertical sync. The zoom port vertical sync pulse has been extended to work with more video controllers.

Testability enhancements

Additional test bits were added to the test registers. This does not affect normal operation.

Header compare logic modified to allow reception of cycle start packets

The PCILynx header compare logic behavior has changed in Rev A. Previously, *always true* comparator values would not receive cycle start packets; however, with Rev A, cycle starts can be received into the GRF. The comparator definition that changed is the Tcode compare enable field (word 0 receive packet compare enable register CMP0_FIELD3_ENABLE [3:0]). See section 5.5.2, *DMA Channel 0 – 4 Word 0 Receive Packet Compare Enable Register @B04 B14 B24 B34 B44*, for the new definition of this field. Any CMP0_FIELD3_ENABLE that is *always true* will receive cycle starts. Selecting only isochronous or asynchronous encoding will not receive cycle starts.

A DMA channel programmed with a CMP0_FIELD3_MASK == 'b0000, which in the original part received everything except cycle start packets, now also receives cycle start packets. This adds a lots of traffic for what may have been intended to be a low traffic channel. The solution is to program the comparator differently.

If one only wants asynchronous packets (but not cycle start packets), not caught by some other higher priority channel, set:

CMP0_FIELD3_ENABLE == 0xA. "normal ASYNC encoding" –> GRF

If one only wants all packets (asynchronous and isochonous, but not cycle start packets) not caught by some other higher priority channel, set:

CMP0_FIELD3_ENABLE == 0x4 Tcode DOES NOT match cmp0_field3 –> GRF and CMP0_FIELD3 == 0x8Tcode == CYCLE_START

SNOOP_ENABLE bit added

A *SNOOP_ENABLE* bit was added to the link layer control register. See section 5.5.6, *1394 Link Layer Control @F04*.

CYCMASTER bit automatically clears

The CYCMASTER bit of the 1394 link layer control @F04 has been modified. This bit will now automatically clear after a 1394 bus reset if the attached PHY is no longer ROOT. The previous version required software to clear this bit after a bus reset.

Local Bus waitstates redefined

The definition of *Waitstates* in the local bus control register has changed. Setting Waitstates == 0 results in a 1 clock access cycle on the local bus, a value of 1, results in a 2 clock access, and so on. See section 5.2.16, *Local Bus Control Register @0B0 {ROM, RAM, AUX, and AV registers}.*

Local Bus AUX_DATA bus

The local bus AUX_DATA bus is set to high-impedance state by PCI reset (unless autoboot is active). This will allow designs that only have zoom port (no AUX/ROM/RAM) to multiplex between various zoom sources without having to add external 3-state buffers. This also requires pull-up/down resistors on the bus if it is not used since the bus is bidirectional.

GPIO polarity control

The definition of the GPIO polarity control bits has changed. See section 4.2.1.5.6, *GPIO Interface*. Also see section 5.2.18, *PCI_GPIO[1:0] Control Register A @0B8* and section 5.2.19, *PCI_GPIO[3:2] Control Register B @0BC*.

Serial EEPROM bit ordering has been corrected

The bit ordering on the autonomous serial EEPROM load was reversed. Rev A and later loads the MS bit of each byte first, and the LS bit last.

Link transmitter modified

The link layer transmitter was modified to only allow an ISO transmit request to be issued to the PHY near the start of a subaction gap. This prevents late-arriving ISO transmit requests from extending into the ASYNC period.

Errata items from previous version corrected

In general, the errata items from the original 12LV21 device were corrected. The exceptions follow:

- Electrical isolation as described in Annex J.6 of IEEE 1394–1995 is not supported by PCILynx.
- PCI performance has been enhanced, however it remains possible to program the PCILynx so that the latencies exceed the PCI 2.1 specification. In particular, the use of local bus waitstates and the aux_rdy signal can cause PCI bus latencies which exceed those specified by PCI 2.1. In some cases a PCI transaction can be held waiting for a local bus transaction to complete. See section 4.2.1.1, *PCI Specification 2.1 Compliance*.
- Separately, a bug in the Rev A PCILynx makes it necessary to set the ENA_POST_WR bit in the miscellaneous control register to 0. Note that Errata for the original PCILynx required that both ENA_POST_WR and ENA_SLV_BURST be set to 1.

Revision B device changes

The following paragraphs describe the Revision B changes to the TSB12LV21A. This device is now known as the TSB12LV21B.

Device ID

Device ID changed from 0x02 to 0x04.

FIFO size

The FIFO size increased from a total of 1 Kbyte to 4 Kbytes. The current RAM is a single 1024 X 33-bit clocked dual-port RAM. Each FIFO is programmable in size from 0 to 256 quadlets (x4). The programmed FIFO size is internally multiplied by 4.

- Allows asynchronous streams (isochronous tcode used in an asynchronous bus phase) in the asynchronous transmit FIFO.
- Corrected receive behavior of certain illegal packets with missing or extra data.
- New ack-type error added: ’h1F packet format error (separate from tcode error)
- Added the subsystem access register at PCI offset 58h. Writes to this register are reflected back to the subsystem ID and subsystem vendor ID registers at PCI offset 2Ch. This provides a method for system BIOS to write values to the subsystem ID and subsystem vendor ID registers and allow them to appear as read-only to the Windows 9x operating system. This method for the subsystem ID and subsystem vendor ID registers can be used when no EEPROM is implemented.

Appendix I Using SNOOP Mode

Set bit 6 SNOOP_ENABLE in the link control register @F04. This forces all received data to DMA channel 0 and ACKs are inhibited.

Only DMA channel 0 should be programmed.

Set up the largest possible receive FIFO.

An extra quadlet is appended on the end of received packets. This quadlet contains the received ACK (ls bits) or 0 for ISO or cycle_start packets.

Appendix J Using the AV Port

ZV_DATA_VALID is an active high signal that indicates when valid data is present on the ZV data bus. It remains high until the back-to-back writes are finished. For example, a typical transaction would be to put a quadlet out to the ZV port. If the ZV port is enabled in 16-bit mode it will require 2 clocks to transfer the 2 doublets to complete the quadlet. ZV_DATA_VALID will remain high for the entire duration of the 2 clocks required to complete the transfer of the quadlet. ZV_PIX_CLK should be used to latch the data. If the interface does not use ZV_DATA_VALID then the ZV port pixel clock should be put into gated mode (local bus control register bit 31 at offset 0B0h). When placed in gated mode ZV_PIX_CLK is only allowed to toggle when data is valid on the ZV data bus. When the ZV port is disabled, the ZV_DATA_VALID signal is 3-stated.

ZV_HSYNC is the video port horizonal sync pulse. Its mode of operation is set using bits 28, 29, and 30 of the local bus control register @0B0h. When these bits are set to 0 it is a special mode where it is assumed the device receiving the data is generating its own HSyncs. An HSync will still be generated once every frame in this mode. When these bits are set to a nonzero value, it is the number of packets between Hsyncs (the number of packets that make up one line of video).

PCILynx PIN	COMMENT
ZV_PIX_CLK	Video port pixel clock (can be inverted via control reg bit) 8-bit mode uses both edges and MUST select divide-by-2 clock (i.e., ext_clk/2, sclk/4, or pci_clk/2)
ZV_VSYNC	Video port vertical sync
ZV_HSYNC	Video port horizonal sync
ZV_DATA_VALID	Video port data valid not used on all chips, if not used must use gated clock mode
AUX_DATA[7:0]	Data output for all bytes for 8-bit mode video data for UV data in 16-bit mode
AUX_DATA[15:8]	Video data for Y data in 16-bit mode

This interface was designed to allow the 1394 digital camera to send packet data to a zoom video interface. For the data sequence and the vertical sync detect logic in the PCILynx to work, the packets (both in the header and the packet payload data) must be received with the big_endian flag set to 0 (which preserves byte addressing).

The pixel clock choices are restricted in 8-bit mode because the data is clocked on every edge (both rising and falling). To keep the controlling state machines synchronized, the clock needs to be divided by 2.