July 2000

National Semiconductor®

Geode™ CS4210 IEEE 1394 OHCI Controller

General Description

The National Semiconductor® Geode™ CS4210 is a PCI-based IEEE 1394 OHCI (Open Host Controller Interface) controller. The CS4210 provides an implementation of the IEEE 1394 Link Layer functionality according to the programming model defined by 1394 OHCI Specification Version 1.0. It supports high speed serial communication up to 400 Mbits per second.

The CS4210 is an implementation of the link layer protocol of the IEEE 1394 high speed serial bus, with additional features to support the transaction and bus management layers. The CS4210 also includes DMA engines for high speed performance data transfer and a host PCI bus interface. Perfect for use in PC, set-top box, thin client, and WebPAD™ system applications.

The CS4210 supports two types of data transfers: asynchronous and isochronous.

The CS4210 provides an external IEEE 1394 physical layer device interface. The CS4210's physical layer interface (PHY-Link) is compatible with the Geode™ CS4103 and other IEEE 1394 physical layer devices.

Features

- Supports 100, 200, and 400 Mbit/sec data transfer rates
- Compliant with 1394 OHCI Specification Version 1.0
- Compatible interface with the Geode CS4103 IEEE P1394a Physical Layer (PHY) device and other IEEE 1394-1995 version and P1394a Draft 2.0 Physical Layer devices
- Eight isochronous transmit contexts
- Eight isochronous receive contexts
- Capable of reading a 128-byte descriptor in one burst
- 128 byte, zero wait state bursting
- Dynamically re-prioritize services
- 2 KB of isochronous transmit FIFO
- 2 KB of asynchronous transmit FIFO
- 4 KB of receiver FIFO
- Per-packet FIFO thresholding
- Four concurrent posted writes
- Eight pending physical responses
- National specific configuration registers
- I^2C interface support for an optional serial EEPROM
- Accepts and generates external 8 kHz reference clock
- 5V tolerant PCI rev 2.1 I/O interface
- 0.25µ CMOS
- 100-pin LQFP (Low-profile Quad Flat Pack) package
- NAND tree for test purposes

System Block Diagram

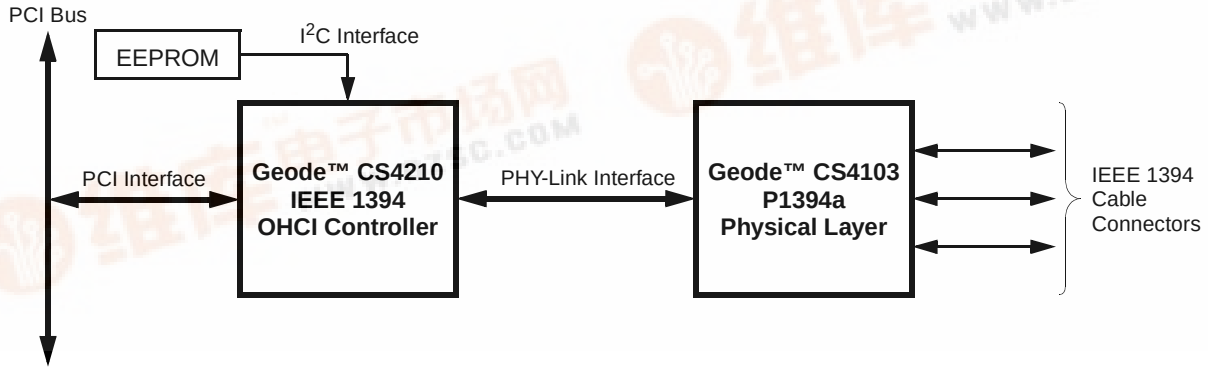


Table of Contents

Table of Contents (Continued)

Table of Contents (Continued)

Table of Contents (Continued)

1.0 Architectural Description

The CS4210 device implements an IEEE 1394 serial bus host controller as specified by the OHCI Specification Version 1.0. It is organized as a collection of reusable modules as illustrated in Figure 1-1.

The interface to the 1394 bus is organized into three clock domains: PCI, SCLK, and SCLK/2. The PCI clock domain can operate at up to 33 MHz. The PCI clock can also be stopped. The SCLK clock domain operates at 49.152 MHz. The SCLK/2 clock domain operates at 24.576 MHz.

1.1 PCI INTERFACE MODULE

The PCI interface module provides a full function bus mastering interface to the PCI bus.

1.2 DMA ENGINE

The DMA engine is decomposed into four functional modules:

- TE: Transfer Engine
  - — Provides generic data movement services to the rest of the DMA engine logic.
- ATDMA: Asynchronous Transmit DMA
  - — Controls the transmission of all asynchronous packets.
- ITDMA: Isochronous Transmit DMA
  - — Controls the transmission of all isochronous packets.
- RDMA: Receive DMA
  - — Processes all received packets (asynchronous, isochronous and physical) and transmit status.

A single port SRAM is shared by the DMA logic for caching control information and descriptor blocks fetched from the host memory. This RAM is used for capturing entire descriptor blocks in a single PCI bus tenure.

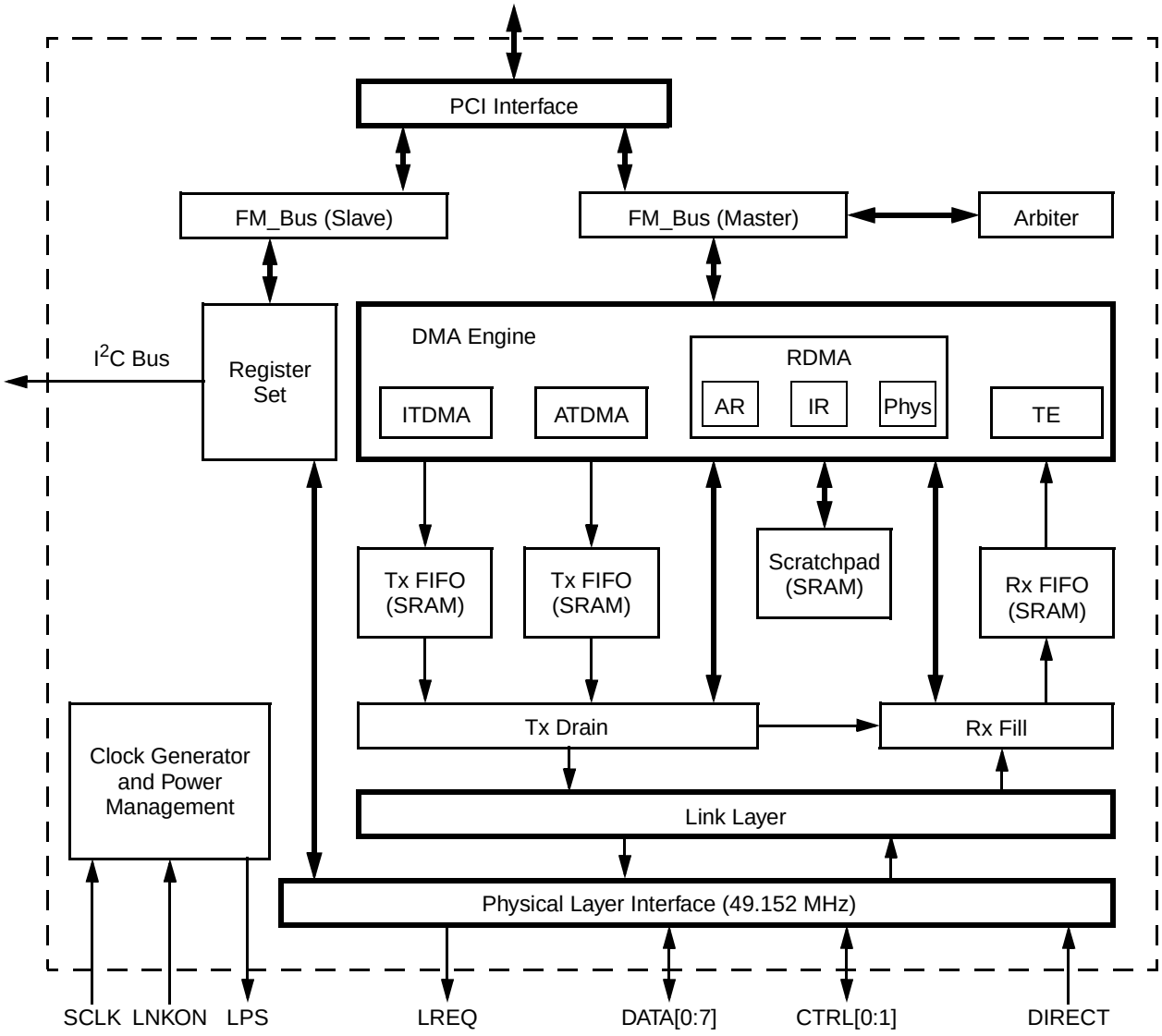


Figure 1-1. Functional Block Diagram

Architectural Description (Continued)

1.2.1 Transfer Engine

The transfer engine performs data movement between different sources/sinks:

1) Host memory (TxFIFOs/RxFIFO, Scratchpad SRAM)

2) DMA modules (ATDMA, ITDMA, or RDMA)

All byte alignment and byte swapping tasks are also performed by the transfer engine.

1.2.2 Host Memory Organization

The 1394 OHCI specification allows for many different data FIFO implementations. This CS4210 implements two transmit FIFOs (asynchronous and isochronous) and a single receive FIFO. The transmit FIFOs share a single embedded dual-port SRAM (36x1024). The receive FIFO uses a single embedded dual-port SRAM (36x1024). A small transmit FIFO is also implemented using latches and some decoding logic.

All FIFOs may be tested using an embedded RAM BIST controller. See Section 4.5.4 "nscRAMBist" on page 94 for more details.

1.2.3 ATDMA

The ATDMA module controls the transmission of all asynchronous packets. This includes AT (Asynchronous Transmit) Request context packets, AT Response context packets and all physical DMA transmit request and response packets. The ATDMA module also controls retransmission of packets as required by the OHCI specification.

1.2.4 ITDMA

The ITDMA module controls the transmission of all isochronous packets. Annex E of the OHCI specification describes the operation of the ITDMA module. This annex was contributed by National Semiconductor.

1.2.5 RDMA

The RDMA module processes all received packets and transmit status. This includes packets destined for the AR (Asynchronous Receive) Request context, AR Response context, all IR (Isochronous Receive) contexts, the Self-ID buffer, and all physical DMA requests (including CSR accesses). It also examines the transmit status to manage the collection of currently active physical DMA requests.

1.3 TRANSMIT DRAIN

The transmit drain module accepts packets from the TxFIFOs (asynchronous and isochronous) and interfaces with the link layer module to transmit these packets. It also places transmit completion status in the TxFIFO.

1.4 RECEIVE FILL

The receive fill module accepts packets from the link layer and places them into the RxFIFO. It performs packet filtering and routing. It also determines which handshake, if any, to return for each received packet.

1.5 LINK LAYER

The link layer module implements a 1394 link layer function developed for this host controller application. It includes support for the CRC32 generation/checking, link state machine, transmit/receive data paths and the generation/reception of cycle start packets. This module includes support for features defined in the P1394a supplement.

1.6 PHYSICAL LAYER INTERFACE

The physical layer interface module implements the external interface to connect to the Geode CS4103 P1394a physical layer device. It includes support for features defined in revision 2.0 of the P1394a specification.

1.7 REGISTER SET

The register set module coordinates slave accesses to the host controller registers. It fields read/write requests from the PCI interface module. It can also read configuration data from a serial EEPROM device via an I^2C interface.

1.8 RELATED DOCUMENTS

The following documents may be useful in understanding the terms and concepts used in this publication.

- 1394 Open Host Controller Interface Specification Release 1.0
- IEEE 1394-1995 High Performance Serial Bus, 1995
- ISO/IEC 13213:1994 Control and Status Register Architecture for Microcomputer Buses International Standards Organization, 1994
- IEEE P1394a Standard for a High Performance Serial bus (Supplement)

2.0 Signal Definitions

This section defines the signals and external interface of the CS4210. Figure 2-1 shows the pins organized by their functional groupings (internal test and electrical pins are not shown).

2.1 PIN ASSIGNMENT

The tables in this section use several common abbreviations. Table 2-1 lists the mnemonics and their meanings.

Figure 2-2 on page 9 shows the pin assignment for the CS4210 with Tables 2-2 and 2-3, on pages 10 and 11, listing the pin assignments sorted by pin number and alphabetically by signal name.

Section 2.2 "Signal Descriptions" starting on page 12 provides a description for each signal within its associated functional group.

Table 2-1. Pin Type Definitions

Mnemonic	Definition
I	Input Pin
I/O	Bidirectional Pin
O	Output
t/s	TRI-STATE Signal
VDD	2.5V Core Power Supply
VDDIO	3.3V I/O Power Supply
VSS	Ground Connection

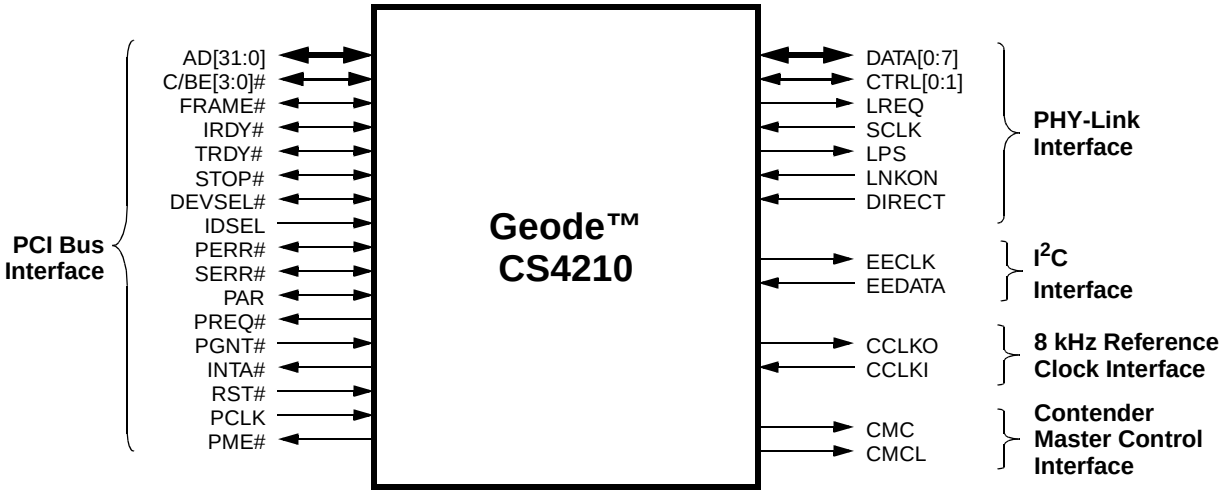


Figure 2-1. Signal Groups

Signal Definitions (Continued)

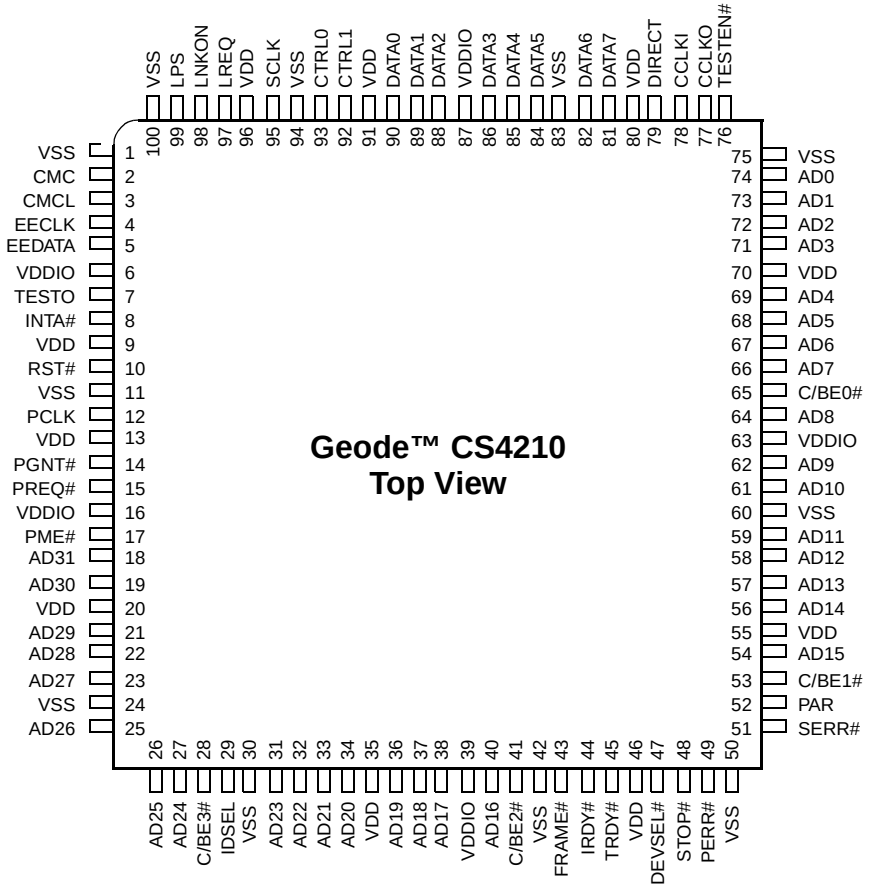


Figure 2-2. Pin Assignment Diagram
Order Number: CS4210VJG

Signal Definitions (Continued)

Table 2-2. Pin Assignment - Sorted by Pin Number

Pin No.	Signal	Type
1	VSS	GND
2	CMC	O
3	CMCL	O
4	EECLK	O
5	EEDATA	I
6	VDDIO	PWR
7	TESTO	O
8	INTA#	O
9	VDD	PWR
10	RST#	O
11	VSS	GND
12	PCLK	I
13	VDD	PWR
14	PGNT#	I
15	PREQ#	O
16	VDDIO	PWR
17	PME#	O
18	AD31	I/O
19	AD30	I/O
20	VDD	PWR
21	AD29	I/O
22	AD28	I/O
23	AD27	I/O
24	VSS	GND
25	AD26	I/O
26	AD25	I/O
27	AD24	I/O
28	C/BE3#	I/O
29	IDSEL	I
30	VSS	GND
31	AD23	I/O
32	AD22	I/O
33	AD21	I/O
34	AD20	I/O
35	VDD	PWR
36	AD19	I/O
37	AD18	I/O
38	AD17	I/O
39	VDDIO	PWR
40	AD16	I/O
41	C/BE2#	I/O
42	VSS	GND
43	FRAME#	I/O
44	IRDY#	I/O
45	TRDY#	I/O
46	VDD	PWR
47	DEVSEL#	I/O
48	STOP#	I/O
49	PERR#	I/O
50	VSS	GND
51	SERR#	I/O
52	PAR	I/O
53	C/BE1#	I/O
54	AD15	I/O
55	VDD	PWR
56	AD14	I/O
57	AD13	I/O
58	AD12	I/O
59	AD11	I/O
60	VSS	GND
61	AD10	I/O
62	AD9	I/O
63	VDDIO	PWR
64	AD8	I/O
65	C/BE0#	I/O
66	AD7	I/O
67	AD6	I/O
68	AD5	I/O
69	AD4	I/O
70	VDD	PWR
71	AD3	I/O
72	AD2	I/O
73	AD1	I/O
74	AD0	I/O
75	VSS	GND
76	TESTEN#	I
77	CCLKO	O
78	CCLKI	I
79	DIRECT	I
80	VDD	PWR
81	DATA7	I/O
82	DATA6	I/O
83	VSS	GND
84	DATA5	I/O
85	DATA4	I/O
86	DATA3	I/O
87	VDDIO	PWR
88	DATA2	I/O
89	DATA1	I/O
90	DATA0	I/O
91	VDD	PWR
92	CTRL1	I/O
93	CTRL0	I/O
94	VSS	GND
95	SCLK	I
96	VDD	PWR
97	LREQ	O
98	LNKON	I
99	LPS	O
100	VSS	GND

Signal Definitions (Continued)

Table 2-3. Pin Assignment - Sorted Alphabetically

Signal	Type	Pin No.
AD0	I/O	74
AD1	I/O	73
AD10	I/O	61
AD11	I/O	59
AD12	I/O	58
AD13	I/O	57
AD14	I/O	56
AD15	I/O	54
AD16	I/O	40
AD17	I/O	38
AD18	I/O	37
AD19	I/O	36
AD2	I/O	72
AD20	I/O	34
AD21	I/O	33
AD22	I/O	32
AD23	I/O	31
AD24	I/O	27
AD25	I/O	26
AD26	I/O	25
AD27	I/O	23
AD28	I/O	22
AD29	I/O	21
AD3	I/O	71
AD30	I/O	19
AD31	I/O	18
AD4	I/O	69
AD5	I/O	68
AD6	I/O	67
AD7	I/O	66
AD8	I/O	64
AD9	I/O	62
C/BE0#	I/O	65
C/BE1#	I/O	53

Signal	Type	Pin No.
C/BE2#	I/O	41
C/BE3#	I/O	28
CCLKI	I	78
CCLKO	O	77
CMC	O	2
CMCL	O	3
CTRL0	I/O	93
CTRL1	I/O	92
DATA0	I/O	90
DATA1	I/O	89
DATA2	I/O	88
DATA3	I/O	86
DATA4	I/O	85
DATA5	I/O	84
DATA6	I/O	82
DATA7	I/O	81
DEVSEL#	I/O	47
DIRECT	I	79
EECLK	O	4
EEDATA	I	5
FRAME#	I/O	43
IDSEL	I	29
INTA#	O	8
IRDY#	I/O	44
LNKON	I	98
LPS	O	99
LREQ	O	97
PAR	I/O	52
PCLK	I	12
PERR#	I/O	49
PGNT#	I	14
PME#	O	17
PREQ#	O	15
RST#	O	10

Signal	Type	Pin No.
SCLK	I	95
SERR#	I/O	51
STOP#	I/O	48
TESTEN#	I	76
TESTO	O	7
TRDY#	I/O	45
VDD	PWR	9
VDD	PWR	13
VDD	PWR	20
VDD	PWR	35
VDD	PWR	46
VDD	PWR	55
VDD	PWR	70
VDD	PWR	80
VDD	PWR	91
VDD	PWR	96
VDDIO	PWR	6
VDDIO	PWR	16
VDDIO	PWR	39
VDDIO	PWR	63
VDDIO	PWR	87
VSS	GND	1
VSS	GND	11
VSS	GND	24
VSS	GND	30
VSS	GND	42
VSS	GND	50
VSS	GND	60
VSS	GND	75
VSS	GND	83
VSS	GND	94
VSS	GND	100

Signal Definitions (Continued)

2.2 SIGNAL DESCRIPTIONS

2.2.1 PCI Bus Interface Signals

Signal Name	Pin No.	Type	Description
AD[31:0]	Refer to Table 2-3	I/O	**Multiplexed Address and Data** AD[31:0] is a physical address during the first clock of a PCI transaction; it is the data during subsequent clocks. When the CS4210 is a PCI master, AD[31:0] are outputs during the address and write data phases, and are inputs during the read data phase of a transaction. When the CS4210 is a PCI slave, AD[31:0] are inputs during the address and write data phases, and are outputs during the read data phase of a transaction.
C/BE[3:0]#	65, 53, 41, 28	I/O	**Bus Command and Byte Enables** Multiplexed bus command and byte enables.
FRAME#	43	I/O	**Cycle Frame** Driven by the initiator to indicate the beginning and duration of an access.
IRDY#	44	I/O	**Initiator Ready** Indicates that the initiator is ready to complete the current data phase of the transaction.
TRDY#	45	I/O	**Target Ready** Indicates that the current data phase of the transaction is ready to be completed.
STOP#	48	I/O	**Stop** Indicates that the current target is requesting the initiator to stop the current transaction.
DEVSEL#	47	I/O	**Device Select** When actively driven, DEVSEL# indicates the driving device has decoded its address as the target of the current access.
IDSEL	29	I	**Initialization Device Select** Used as a chip select during configuration read and write transactions.
PERR#	49	I/O	**Parity Error** Used for reporting data parity errors during all PCI transactions except a Special Cycle.
SERR#	51	I/O	**System Error** Used for reporting address parity errors, data parity errors on the Special Cycle command, or any other system error where the result will be catastrophic.
PAR	52	I/O	**Parity** PAR is even parity across AD[31:0] and C/BE[3:0]. PAR is an input when AD[31:0] are inputs and is an output when AD[31:0] are outputs.
PREQ#	15	O	**PCI Bus Request** PCI bus request to PCI bus arbiter.
PGNT#	14	I	**PCI Bus Grant** PCI bus grant from PCI bus arbiter.
INTA#	8	O	**Interrupt A** 1394 OpenHCI PCI interrupt.

Signal Definitions (Continued)

2.2.1 PCI Bus Interface Signals (Continued)

Signal Name	Pin No.	Type	Description
RST#	10	I	**PCI Reset** RST# is driven low to reset the device.
PCLK	12	I	**Clock** 0-33 MHz PCI clock.
PME#	17	O	**Power Management Event** PCI power management pin as defined in the PCI Bus Power Management Specification Revision 1.1.

2.2.2 PHY-Link Interface Signals

Signal Name	Pin No.	I/O	Description
DATA[0:7]	81, 82, 84, 85, 86, 88, 89, 90	I/O	**PHY Data** Bidirectional data lines driven by both the Link and PHY layer modules. The width of the data bus depends on the speed of data transfer rate. Packet rate for 100 Mbit/sec transfers use DATA[0:1], 200 Mbit/sec transfers use DATA[0:3], 400 Mbit/sec transfers use DATA[0:7]. **Note:** DATA0 is considered the MSB (most significant bit) based upon the IEEE 1394-1995 specification.
CTRL[0:1]	93, 92	I/O	**Control bits 1 and 0** Bidirectional handshaking signals driven by both the Link and PHY layer modules. The CS4210 and CS4103 use these signals to arbitrate the control of the PHY-Link interface. The control bits also indicate the type of transfer communicating between the two layers namely idle, status, receive, and transmit.
LREQ	97	O	**Link Request** Used by the CS4210 to request access of the 1394 bus and to read/write the internal registers of the CS4103.
SCLK	95	I	**Sync Clock** The 49.152 MHz clock input driven by the CS4103's PLL block synchronized to the 1394 bus clock. This clock is also used to synchronize the LREQ, CTRL[0:1], and DATA[0:7] communication protocol between the CS4210 and CS4103.
LPS	99	O	**Link Power Status** Indicates the power status of the CS4210. If LPS is low indicating the CS4210 is not powered, the signals CTRL[0:1], DATA[0:7], and SCLK connected to the CS4210 are disabled.
LNKON	98	I	**Link On** Indicates to the CS4210 that the CS4103 has received a Link-On packet addressed to this node.
DIRECT	79	I	**Direct** High indicates direct connection. Low indicates isolation barrier. Set high when using the single capacitor bus hold isolation.

Signal Definitions (Continued)

2.2.3 Miscellaneous Interface Signals

Signal Name	Pin No.	Type	Description
EECLK	4	O	**Serial EEPROM Clock** The I^2C bus clock signal.
EEDATA	5	I	**Serial EEPROM Data** The I^2C data signal.
CCLKO	77	O	**Cycle Clock Output** The 8 kHz reference clock output.
CCLKI	78	I	**Cycle Clock Input** The 8 kHz reference clock input.
CMC	2	O	**Contender Master Control** This output is set via the nscCMCControl.CMC bit (BAR1+Offset 18h[0]). The value placed on the bit is directly reflected on this pin.
CMCL	3	O	**Contender Master Control Link Enabled** This output is set via the nscCMCControl.CMCL bit (BAR1+Offset 18h[1]). The value of the bit is reflected on the output when HCControl.linkEnable (BAR0+Offset 50h[17]) is set. Otherwise it is 0.
TESTO	7	O	**Test Out** National internal test pin, user must float.
TESTEN#	76	I	**Test Enable** National internal test pin, user must tie high.

2.2.4 Power Supplies and Ground Connections

Signal Name	Pin No.	Type	Description
VDD	9, 13, 20, 35, 46, 55, 70, 80, 91, 96	PWR	**2.5V Core Power Supply Connections (Total of 10)**
VDDIO	6, 16, 39, 63, 87	PWR	**3.3V I/O Power Supply Connections (Total of 5)**
VSS	1, 11, 24, 30, 42, 50, 60, 75, 83, 94, 100	GND	**Ground Connections (Total of 11)**

3.0 Operational Description

3.1 OVERVIEW

The CS4210 is an implementation of the link layer protocol of the 1394 serial bus, with additional features to support the transaction and bus management layers. The CS4210 also includes DMA engines for high-performance data transfer and a PCI host bus interface. IEEE 1394 serial bus (and 1394 OpenHCI) protocols support two types of data transfer: asynchronous and isochronous.

- Asynchronous data transfer puts the emphasis on guaranteed delivery of data, with less emphasis on guaranteed timing.
- Isochronous data transfer is the opposite, with the emphasis on the guaranteed timing of the data, and less emphasis on delivery.

3.1.1 Asynchronous Data Transfer Functions

The CS4210 can transmit and receive all of the defined 1394 packet formats. Packets to be transmitted are read out of host memory and received packets are written into host memory, both using DMA. The CS4210 can also be programmed to act as a bus bridge between the host bus and 1394 devices by directly executing 1394 read and write requests as reads and writes to the host bus memory space.

3.1.2 Isochronous Data Transfer Functions

The CS4210 is capable of performing the cycle master function as defined by the IEEE 1394 OHCI specification. This means it contains a cycle timer and counter, and can queue the transmission of a special packet called a “cycle start” after every rising edge of the 8 kHz cycle clock. The CS4210 can generate the cycle clock internally or use an external reference connected to the CCLKI input (pin 78). When not the cycle master, the CS4210 keeps its internal cycle timer synchronized with the cycle master node by correcting its own cycle timer with the reload value from the cycle start packet. Conceptually, the CS4210 supports one DMA controller each for isochronous transmit and isochronous receive. The CS4210 provides eight isochronous transmit contexts. The isochronous transmit DMA controller can transmit from each context during each cycle. Each context can transmit data for a single isochronous channel. The CS4210 provides eight isochronous receive contexts. The isochronous receive DMA controller can receive data for each context during each cycle. Each context can be configured to receive data from a single isochronous channel. Additionally, one context can be configured to receive data from multiple isochronous channels (see bit 28, multiChanMode, in Table 4-53 on page 87 for programming details).

3.1.3 Miscellaneous Functions

Upon detecting a bus reset, the CS4210 automatically flushes all packets queued for asynchronous transmission. Asynchronous packet reception continues without interruption, and a token appears in the received request packet stream to indicate the occurrence of the bus reset. When the CS4103 provides the new local node ID, the CS4210 loads this value into its Node ID register, see Table 3-1. Asynchronous packet transmit will not resume until directed to by software. Because target node ID values may have changed during the bus reset, software will not generally be able to re-issue old asynchronous requests until software has determined the new target node IDs. Isochronous transmit and receive functions are not halted by a bus reset, instead they restart as soon as the bus initialization process is complete. A number of management functions are also implemented by the CS4210. A global unique ID register, shown in Table 3-2, can only be written once. For full compliance with higher level standards, this register must be written before the boot block is read. To make this implementation simpler, the CS4210 has an interface to an external serial I^2C EEPROM such as the Fairchild Semiconductor NM24C02. The CS4210 also supports four registers that implement the compare-swap operation needed for isochronous resource management.

Table 3-1. BAR0+Offset E8h: Note ID and Status Register Map

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
iDValid	root	RSVD		CPS	RSVD											busNumber										nodeNumber					

Table 3-2. GUID Register Map

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BAR0+Offset 24h-27h GUIDHi Register **BAR0+Offset 28h-2Bh GUIDLo Register**																															
node_vendor_ID																								chip_ID_Hi							
chip_ID_Lo																															

Operational Description (Continued)

3.2 SOFTWARE INTERFACE OVERVIEW

There are three basic means by which software communicates with the CS4210: registers, DMA, and interrupts.

3.2.1 Registers

The host architecture (PCI, for example) is responsible for mapping the CS4210's registers into a portion of the host's address space.

3.2.2 DMA Operation

DMA transfers in the CS4210 are accomplished through one of two methods: DMA memory and physical response DMA.

3.2.2.1 DMA Memory

DMA memory resident data structures are used to describe lists of data buffers. The CS4210 automatically sequences through this buffer descriptor list. This data structure also contains status information regarding the transfers. Upon completion of each data transfer, the DMA controller conditionally updates the corresponding DMA context command and conditionally interrupts the processor so it can observe the status of the transaction. A set of registers within the CS4210 is used to initialize each DMA context and to perform control actions such as starting the transfer.

3.2.2.2 Physical Response DMA

The CS4210 can be programmed to accept 1394 read and write transactions as reads and writes to host memory space. In this mode, the CS4210 acts as a bus bridge from the 1394 bus into host memory. The formats for the data sent and received in all these modes are specified in the 1394 Open Host Controller Interface Specification Release 1.00.

3.2.3 Interrupts

When any DMA transfer completes (or aborts), an interrupt may be sent to the host system. In addition to the interrupt sources which correspond to each DMA context completion, there is also a set of interrupts which correspond to other CS4210 functions/units. For example, one of these interrupts could be sent when a Self-ID packet stream has been received. The processor interrupt line is controlled by the IntEvent and IntMask registers. The IntEvent register indicates which interrupt events have occurred, and the IntMask register is used to enable selected interrupts. Software writes to the IntEventClear register to clear interrupt conditions in IntEvent. In addition, there are registers used by the isochronous transmit and isochronous receive controllers to indicate interrupt conditions for each context.

Table 3-3 shows a map of the IntEvent and IntMask Set/Clear registers. Refer to Section 4.4.16.1 "IntEvent Register" on page 70 and Section 4.4.16.3 "IntMask Register" on page 72 for further information details.

3.2.3.1 Asynchronous Transmit Interrupts

Each asynchronous DMA context has one interrupt indication bit in the IntEvent register. For requests, it is the reqTxComplete bit and for responses it is the respTxComplete bit. This interrupt indication bit is set to one if a completed OUTPUT_LAST command has the "i" field set to 11b, or if the "i" field is set to 01b and transmission of the packet did not yield an ack_complete or an ack_pending.

3.2.3.2 Asynchronous Receive Interrupts

There are two interrupts for each context (request and response) that software can use to gauge the usage of the receive buffers. If software needs to be informed of the arrival of each packet being sent to the context buffers, it can use the RQPkt or RSPkt interrupts in the IntEvent register. If software needs to be informed of the completion of a buffer, it can set the descriptor i field to 11b, which triggers either the ARRQ or ARRS interrupt in the IntEvent register.

Table 3-3. IntEvent and IntMask Register Map

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BAR0+Offset 80h **BAR0+Offset 84h**					**IntEvent Set Register** **IntEvent Clear Register**																											
RSVD					phyRegRcvd	cycleTooLong	unrecoverableError	cycleInconsistent	cycleLost	cycle64Seconds	cycleSynch	phy	RSVD	busReset	selfIDcomplete	RSVD							lockRespErr	postedWriteErr	isochRx	isochTx	RSPkt	RQPkt	ARRS	ARRQ	respTxComplete	reqTxComplete
BAR0+Offset 88h **BAR0+Offset 8Ch**					**IntMask Set Register** **IntMask Clear Register**																											
masterIntEnable	RSVD				phyRegRcvd	cycleTooLong	unrecoverableError	cycleInconsistent	cycleLost	cycle64Seconds	cycleSynch	phy	RSVD	busReset	selfIDcomplete	RSVD							lockRespErr	postedWriteErr	isochRx	isochTx	RSPkt	RQPkt	ARRS	ARRQ	respTxComplete	reqTxComplete

Operational Description (Continued)

3.2.3.3 Isoch Tx and Rx Context Interrupts

Each of the eight implemented isochronous transmit and each of the eight implemented isochronous receive contexts can generate an interrupt. Software can enable interrupts on a per-context basis by setting the corresponding IsochTxnContextIntMask or IsochRxnContextIntMask bit to one. To efficiently handle interrupts which could conceivably be generated from eight different contexts in close proximity to one another, there is a single bit for all IT DMA contexts and another for all IR DMA contexts in the CS4210 IntEvent register. These bits signify that at least one but potentially several IT or IR DMA contexts attempted to generate an interrupt. Software can read the isochTxIntEvent register to find out which isochronous transmit context(s) are involved. Software can read the IsochRxIntEvent register to find out which isochronous receive context(s) are involved.

Table 3-4 shows a map of the IsochTx/Rx Context Interrupt Event and Mask Set/Clear registers. Refer to Section 4.4.16.4 on page 73 through Section 4.4.16.7 on page 74 for further register information.

The number of supported isochronous DMA contexts varies for 1394 OHCI implementations from a minimum of four to a maximum of 32. Software can determine the number of supported IT or IR DMA contexts by writing FFFF_FFFFh to IsochTxIntMask register for IT and IsochRxIntMask register for IR, and then reading it back. Bits returned as 1's indicate supported contexts, and bits returned as 0's indicate unsupported/unimplemented contexts.

Table 3-4. IsochTx and IsochRx Context Interrupt Related Registers

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BAR0+Offset 90h **BAR0+Offset 94h**																								**IsochTxIntEvent Set Register** **IsochTxIntEvent Clear Register**							
RSVD																								isochTxInt7	isochTxInt6	isochTxInt5	isochTxInt4	isochTxInt3	isochTxInt2	isochTxInt1	isochTxInt0
BAR0+Offset 98h **BAR0+Offset 9Ch**																								**IsochTxIntMask Set Register** **IsochTxIntMask Clear Register**							
RSVD																								isochTxIntMask7	isochTxIntMask6	isochTxIntMask5	isochTxIntMask4	isochTxIntMask3	isochTxIntMask2	isochTxIntMask1	isochTxIntMask0
BAR0+Offset A0h **BAR0+Offset A4h**																								**IsochRxIntEvent Set Register** **IsochRxIntEvent Clear Register**							
RSVD																								isochRxInt7	isochRxInt6	isochRxInt5	isochRxInt4	isochRxInt3	isochRxInt2	isochRxInt1	isochRxInt0
BAR0+Offset A8h **BAR0+Offset ACh**																								**IsochRxIntMaskSet Register** **IsochRxIntMaskClear Register**							
RSVD																								isochRxIntMask7	isochRxIntMask6	isochRxIntMask5	isochRxIntMask4	isochRxIntMask3	isochRxIntMask2	isochRxIntMask1	isochRxIntMask0

Operational Description (Continued)

When the IntEvent.cycleInconsistent condition occurs, the IT and IR DMA controllers continue processing running contexts normally, except that contexts with the Context-Control.cycleMatchEnable bit set remain inactive and cycleMatch processing is, in effect, disabled. To re-enable cycleMatch processing, software must first stop the IT and/or IR contexts for which cycleMatch is enabled (by clearing ContextControl.run and waiting for ContextControl.active to clear, then clearing the IntEvent.cycleInconsistent interrupt (read BAR0+Offset 84h[23]). The stopped IR contexts may then be started. The stopped IT contexts may also be started, but software should not schedule any transmits to occur for these contexts for at least two cycles immediately following the clearing of the interrupt condition.

Table 3-5 is a register format for the eight Isochronous Transmit Context Control Set/Clear registers. Refer to Section 4.4.26.1 "Isoch Transmit Context Control Register" on page 84 for further register information.

Table 3-5. IsochTx[7:0]ContextControl Set/Clear Register Formats

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IsochTxnContextControl Set Register IsochTxnContextControl Clear Register																															
cycleMatchEnable	cycleMatch															run	RSVD		wake	dead	active	RSVD					event code				

Operational Description (Continued)

3.3 COMMON DMA CONTROLLER FEATURES

The CS4210 provides several types of DMA functionality:

- General-purpose DMA handling asynchronous transmit and receive packets and isochronous transmit and receive packets.
- An inbound bus bridge function that allows 1394 devices to directly access system memory called “physical DMA.”
- A separate write buffer for the received Self-ID packets.
- A mapping between a 1 KB block in system memory and the first 1K of configuration ROM.

3.3.1 Context Registers

A context provides the basic information to the CS4210 to allow it to fetch and process descriptors for one of the several DMA controllers. All contexts (except for Self-ID) have a ContextControl register and a CommandPtr register. The format of the ContextControl Registers is DMA controller specific.

Table 3-6 is a register format of the Contex.Control and CommandPtr registers. Refer to Section 4.4.24 starting on page 80 through Section 4.4.27 for further register information.

3.3.2 ContextControl.event

The packet event codes shown in Table 3-7 on page 20 are possible values for the five-bit ContextControl.*event* field. This field may contain either a 1394 defined ack code or an OpenHCI generated event code. Bits [15:0] of the ContextControl register may be written into host memory to indicate packet and/or DMA descriptor status. However, all possible event codes which may appear in a particular context’s ContextControl register may not necessarily ever be written into host memory for a packet or DMA descriptor status, depending on circumstances and the functionality of the context. The list of ack codes provided in Table 3-7 is informative not normative (i.e., for asynchronous packets the event code may be set to any ack code specified in current and future 1394 standards). OpenHCI generated event codes have an “evt_” prefix and are denoted by a code with the high (fifth) bit equal to 0. In some cases for isochronous I/O OpenHCI may generate a 1394 style ack code for ContextControl.event.

Table 3-6. ContextControl and CommandPtr Registers Formats

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AsyncReqTxContextControl Set/Clear Registers **AsyncRespTxContextControl Set/Clear Registers**																															
RSVD																run	RSVD		wake	dead	active	RSVD						event code			
AsyncReqRxContextControl Set/Clear Registers **AsyncRespRxContextControl Set/Clear Registers**																															
RSVD																run	RSVD		wake	dead	active	RSVD		spd				event code			
IsochTxnContextControl Set/Clear Registers																															
cycleMatchEnable	cycleMatch															run	RSVD		wake	dead	active	RSVD						event code			
IsochRxnContextControl Set/Clear Registers																															
bufferFill	isochheader	CycleMatchEnable	multiChanMode	RSVD												run	RSVD		wake	dead	active	RSVD		spd				event code			
CommandPtr Register																															
descriptorAddress																														Z	

Operational Description (Continued)

Table 3-7. Packet Event Codes

Code	Name	DMA	Description
00h	evt_no_status	AT, AR, IT, IR	No event status.
001h	Reserved	--	Reserved
02h	evt_long_packet	IR	The received data length was greater than the buffer's data_length.
03h	evt_missing_ack	AT	A subaction gap was detected before an ack arrived or the received ack had a parity error.
04h	evt_underrun	AT, IT	Underrun on the corresponding FIFO. The packet was truncated.
05h	evt_overrun	IR	A receive FIFO overflowed during the reception of an isochronous packet.
06h	evt_descriptor_read	AT, AR, IT, IR	An unrecoverable error occurred while the CS4210 was reading a descriptor block.
07h	evt_data_read	AT, IT	An error occurred while the CS4210 was attempting to read from host memory in the data stage of descriptor processing.
08h	evt_data_write	AR, IR, IT	An error occurred while the CS4210 was attempting to write to host memory either in the data stage of descriptor processing (AR, IR), or when processing a single 16-bit host memory write (IT).
09h	evt_bus_reset	AR	Identifies a CS4103 packet in the receive buffer as being the synthesized bus reset packet. (See Section 3.8 "Bus Resets" on page 30.)
0Ah	evt_timeout	AT	Indicates that the asynchronous transmit response packet expired and was not transmitted.
0Bh	evt_tcode_err	AT, IT	A bad tCode is associated with this packet. The packet was flushed.
0Ch-0Dh	Reserved	--	Reserved
0Eh	evt_unknown	AT, AR, IT, IR	An error condition has occurred that cannot be represented by any other event codes defined herein.
0Fh	evt_flushed	AT	Sent by the link side of the output FIFO when asynchronous packets are being flushed due to a bus reset.
10h	Reserved	---	Reserved for definition by future 1394 standards.
11h	ack_complete	AT, AR, IT, IR	For asynchronous request and response packets, this event indicates the destination node has successfully accepted the packet. If the packet was a request subaction, the destination node has successfully completed the transaction and no response subaction follows. The event code for transmitted CS4103, isochronous, asynchronous stream and broadcast packets, none of which yields a 1394 ack code, are set by hardware to ack_complete unless an event occurs.
12h	ack_pending	AT, AR	The destination node has successfully accepted the packet. If the packet was a request subaction, a response subaction follows at a later time. This code is not returned for a response subaction.
13h	Reserved	---	Reserved for definition by future 1394 standards.
14h	ack_busy_X	AT	The packet could not be accepted after max ATRetries (see Section 4.4.3 "ATRetries Register" on page 59) attempts, and the last ack received was ack_busy_X.
15h	ack_busy_A	AT	The packet could not be accepted after max ATRetries (see Section 4.4.3 "ATRetries Register" on page 59) attempts, and the last ack received was ack_busy_A.
16h	ack_busy_B	AT	The packet could not be accepted after max AT Retries (see Section 4.4.3 "ATRetries Register" on page 59) attempts, and the last ack received was ack_busy_B.
17h-1Ah	Reserved		Reserved for definition by future 1394 standards.
1Bh	ack_tardy	AT	The destination node could not accept the packet because the link and higher layers are in a suspended state.
1Ch	Reserved	---	Reserved for definition by future 1394 standards.
1Dh	ack_data_error	AT, IR	The destination node could not accept the block packet because the data field failed the CRC check, or because the length of the data block payload did not match the length contained in the data_length field. This code is not returned for any packet that does not have a data block payload.
1Eh	ack_type_error	AT, AR	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).
1Fh	Reserved	---	Reserved for definition by future 1394 standards.

Operational Description (Continued)

3.3.2.1 ContextControl.run

The ContextControl.run bit is set by software when the CS4210 is to begin processing descriptors for the context. Before software sets ContextControl.run, ContextControl.active must not be set, and the CommandPtr register for the context must contain a valid descriptor block address and a Z value that is appropriate for the descriptor block address. Software may stop the CS4210 from further processing of a context by clearing ContextControl.run. When a ContextControl.run is cleared, the CS4210 will stop processing of the context in a manner that will not impact the operation of any other context or DMA controller. The CS4210 may require a significant amount of time to safely stop processing for a context but when the CS4210 does stop, it clears ContextControl.active. If software clears a ContextControl.run for an isochronous context while the CS4210 is processing a packet for the context, the CS4210 continues to receive or transmit the packet and update descriptor status. The CS4210 does, however, stop at the conclusion of that packet. If ContextControl.run is cleared for a non-isochronous context, the CS4210 may stop processing at any convenient point as long as the context and descriptors end up in a consistent state (e.g., status updated if a packet was sent and acknowledged). Clearing ContextControl.run may have other side effects that are DMA controller dependent. These effects are described in the subsections of Section 4.4 "OHCI Configuration Registers" starting on page 48 that cover each of the DMA controllers. When software clears ContextControl.run and the CS4210 has stopped, the CS4210 is not necessarily in a state that can be restarted simply by setting ContextControl.run. Software should always ensure that CommandPtr.descriptorAddress and CommandPtr.Z are set to valid values before setting ContextControl.run.

3.3.2.2 ContextControl.wake

When software adds to a list of descriptors for a context, the CS4210 may have already read the descriptor that was at the end of the list before it was updated. The value that the CS4210 read may contain a Z value of zero indicating the end of the descriptor list. The ContextControl.wake bit provides a simple semaphore to the hardware to indicate that the list may have changed since the last time the CS4210 read a descriptor. Therefore, if the CS4210 had fetched a descriptor and the indicated branch address had a Z value of zero, then the CS4210 rereads the pointer value.

For transmit contexts and receive contexts in buffer-fill mode (a mode in which a context can receive multiple packets into one data buffer), if the Z value is still zero, then the end of the list was reached and the CS4210 clears ContextControl.active. For receive contexts in buffer-fill mode, if the Z value is still zero on the reread, then the packet cannot be accepted.

For asynchronous contexts, the CS4210 returns the appropriate ack_busy* code. In addition, the CS4210 “backs out” the packet by not updating the buffer’s byte count (resCount), and flushes the packet from the FIFO. The CS4210 does not go inactive, as there is still buffer space available and software is attempting to provide more buffer space. For both transmit and receive contexts, if the Z value is now non-zero, the CS4210 continues processing. In order to ensure that a wake condition is not missed, the CS4210 clears ContextControl.wake before it reads or rereads a descriptor. ContextControl.wake is ignored when ContextControl.run is zero.

3.3.2.3 ContextControl.active

ContextControl.active is set and cleared only by the CS4210. It is set when the CS4210 receives an indication from software that a valid descriptor is available for processing. This indication occurs as a result of software setting the ContextControl.run or by software setting ContextControl.wake while ContextControl.run is set. There are four cases in which the CS4210 clears ContextControl.active:

1) When a branch is indicated by a descriptor but the Z value of the branch address is 0.
2) When software clears ContextControl.run and the CS4210 has reached a safe stopping point.
3) While ContextControl.dead is set.
4) After a hardware or software reset of the CS4210.

Additionally, for the asynchronous transmit contexts (request and response), the CS4210 clears ContextControl.active when a bus reset occurs. When ContextControl.active is cleared and ContextControl.run is already clear, the CS4210 sets the IntEvent bit for the context. This interrupt is the same interrupt that would have been generated by the context if a completed descriptor had indicated that an interrupt should be generated.

3.3.2.4 ContextControl.dead

ContextControl.dead is used to indicate a fatal error in processing a descriptor. When ContextControl.dead is set by the CS4210, ContextControl.active is immediately cleared but ContextControl.run remains set. In addition, setting ContextControl.dead causes an unrecoverableError interrupt event and blocks a normal context event interrupt from being set. ContextControl.dead is immediately cleared when software clears ContextControl.run or by either a hardware or software reset of the CS4210. Software can determine the cause of a context going dead by checking the ContextControl.event code. The defined reasons for the CS4210 to set ContextControl.dead are described in Section 3.7 "Host Bus Errors" on page 28.

Operational Description (Continued)

3.3.2.5 CommandPtr

Software initializes CommandPtr.descriptorAddress to contain the address of the first descriptor block that the CS4210 accesses when software enables the context by setting ContextControl.run. Software also initializes CommandPtr.Z to indicate the number of descriptors in the first descriptor block. Software only writes to this register when both ContextControl.run and ContextControl.active are zero. The CS4210's behavior when this rule is violated is undefined. Since the CS4210 utilizes the CommandPtr register while processing a context, there is a set of guidelines by which software may safely and deterministically read CommandPtr. These guidelines are based on the ContextControl bits as listed in Table 3-8 (X = don't care).

If ContextControl.run is set and ContextControl.dead is not set, then the contents of CommandPtr are only specified if both ContextControl.active and ContextControl.wake are clear. In this instance, CommandPtr.descriptorAddress contains the address of a descriptor within the last descriptor block that was executed. If ContextControl.run and ContextControl.dead are both set, then descriptorAddress points to a descriptor within the descriptor block in which an unrecoverable error occurred. Except for the case where software initializes CommandPtr, the value of CommandPtr.Z is undefined and Z may contain a value that is implementation dependent. The value of CommandPtr is undefined after a hardware or software reset of the CS4210. When software sets ContextControl.run to 1 and CommandPtr.Z contains an invalid value for the controller and context, or if a Z value is invalid for a fetched descriptor block in a running context, the CS4210: sets ContextControl.dead to 1 and sets ContextControl.event to evt_unknown and will not process any descriptors in that context.

Table 3-8. CommandPtr Read Values

ContextControl Bits				CommandPtr.descriptor Address Value
run	dead	active	wake	
0	0	0	X	A descriptor block address. Either last written or last executed.
0	0	1	X	Contents unspecified.
1	0	0	0	Refers to the descriptor block that contains the Z = 0 that caused the CS4210 to set active to 0.
1	0	0	1	Contents unspecified.
1	0	1	0	Contents unspecified.
1	0	1	1	Contents unspecified.
1	1	0	X	Points to the descriptor block in which a fatal error occurred.

Operational Description (Continued)

3.4 LIST MANAGEMENT

All contexts use an identical method for controlling the processing of descriptors associated with the context. This presents a uniform interface to controlling software and allows reuse of hardware on the CS4210.

3.4.1 Context Initialization

Software initializes the context by first checking to see that ContextControl.run, ContextControl.active, and ContextControl.dead are all 0. Then, CommandPtr.descriptorAddress is written to point to a valid descriptor block and CommandPtr.Z is set to a value that is consistent with the descriptor block. Then ContextControl.run can be set.

3.4.2 Appending to Running List

Software may append to a list of descriptors at any time. Software may append either a single descriptor or a linked list of descriptors. When the to-be-appended list is properly formatted, software updates the branch address and Z value of the descriptor that was at the end of the list being processed by the CS4210. When software completes the linking process it must set ContextControl.wake for the context. This ensures that the CS4210 resumes operation if it had previously reached the end of the list and gone inactive.

3.4.3 Stopping a Context

Software can stop a running context by clearing ContextControl.run. The context might not stop immediately. To ensure that the context has stopped, software must wait for ContextControl.active to be cleared by the CS4210. This indicates that the CS4210 has completed all processing associated with the context.

3.4.4 Hardware Behavior

The CS4210 has several DMA controllers each of which has one or more contexts. Each DMA controller is expected to examine each of its contexts on a periodic basis and make operational decisions based on the context state as contained in ContextControl. The DMA controller examines the state of the active, run, wake, and dead bits to govern descriptor processing. This process is executed once each time a context is ‘scheduled’. Scheduling of a context is dependent on the DMA controller. For example, an isochronous transmit context is scheduled once per cycle while an asynchronous request transmit context is only scheduled once per fairness interval.

3.5 ASYNCHRONOUS RECEIVE

The CS4210 accepts 1394 transactions and groups them as follows:

Physical Requests - Physical requests, including physical read, physical write, and lock requests to some CSR registers (see Section 4.4.4 "Autonomous CSR Resources" on page 60) are handled directly by the CS4210 and are not made visible to system software. The CS4210 uses a dedicated physical response unit to handle these requests. This unit will not block processing of other transaction types while dealing with physical requests. Section 3.6 "Physical Requests" on page 26 provides details on which requests can be processed as physical.

Self-ID Packets - CS4103 packets with the Self-ID format can be received at any time. However, only those packets that are received during the Self-ID phase of bus initialization which immediately follows a bus reset are considered to be Self-ID packets. Others are considered simply to be PHY packets which are handled like asynchronous requests. The CS4210 can be programmed to accept or ignore Self-ID packets. When Self-ID packets are accepted, they are stored in a special memory buffer which has a dedicated controller and context. Because of this special memory buffer, Self-ID packets can never get ‘stuck’ in a FIFO.

Asynchronous Responses - When the host system initiates a request through the asynchronous transmit request context, the response is handled by the asynchronous receive response context. The fact that host system software initiates the process and the fact that the CS4210 has a separate context for responses, allows system software to budget for all responses which ensures that the CS4210 always has a place in system memory to store a response when it arrives. In the unlikely event that the CS4210 does not have a place for the response it is allowed to drop the response when it arrives. This causes a split-transaction time-out which is an error condition with which the software is already able to deal.

Asynchronous Requests - A request may arrive at the CS4210 at any time. Additionally, a request can be of any size up to the limits imposed by the max_rec field in the Bus_Info_Block (see Section 4.4.7 "Bus Options Register" on page 62). Due to the unpredictable nature of this transaction type, it is impractical for the system software to ensure that there is always sufficient buffer space defined in the asynchronous request receive buffers. If the FIFO which is receiving requests becomes full, all subsequent requests are busied until there is room to receive them.

Operational Description (Continued)

3.5.1 Unrecoverable Error

If an unrecoverable error occurs when the CS4210 is writing to the AR DMA request buffer, a fail indication is sent to the link side of the FIFO. This indicates that the link side should set its count to zero which will busy further read requests and write requests that are destined for the AR DMA request buffer. If the AR DMA request context has an unrecoverable error, the system side of the FIFO continues to unload the FIFO even though the AR DMA request context is dead. All asynchronous requests that would have been sent to the AR DMA request queue are dropped and no responses for them are sent to the initiating node. Dropping requests destined for the AR DMA request queue is acceptable because:

1) AR DMA read requests are always split transactions (ack_pended),
2) write requests within the physical range have been ack_pended and
3) write requests above the physical range which have been posted (ack_completed) are by definition permitted to fail.

3.5.2 Ack Codes for Write Requests

For write requests that are handled by the physical request controller, the CS4210 may send an ack_complete before the data is actually written to system memory. For a full description of which requests are candidates for physical requests, refer to Section 3.6 "Physical Requests" on page 26. The ack_code sent for write requests to offsets in the range of 0000_FFFF_FFFFh to FFFE_FFFF_FFFFh when not busied is always ack_complete. The ack_code sent for requests to offsets in the range FFFF_0000_0000h to FFFF_FFFF_FFFFh and for block requests with a non-zero extended tcode is always ack_pending.

3.5.3 Posted Writes

As described above, a write request that is handled by the physical request controller or which is in the address range 0000_FFFF_FFFFh to FFFE_FFFF_FFFFh to be handled by the asynchronous request unit, may generate an ack_complete before the data is actually written to the designated system memory location. These writes are referred to as posted writes. Write requests to the physical memory range of the host may be posted if software has enabled posted writes (see Section 4.4.10 "PostedWriteAddress Register" on page 64). If posting is not enabled, the CS4210 will not return a complete indication (ack_complete or resp_complete) until the data has been successfully written to the addressed location in physical memory. If posting of physical writes is enabled, then the CS4210 is allowed to return ack_complete to a physical write request with certain restrictions. This CS4210 supports four posted writes. However, for error reporting purposes a posted write is considered pending until the write is actually completed to the offset address. For each pending posted write, there is an error reporting register to hold the request's source node ID and 48-bit offset address should that posted write fail. If the maximum allowed posted writes are pending, the CS4210 must return either ack_pending or ack_busy* for subsequent posted write request candidates and only return resp_complete when those writes have actually been performed. Read and write requests within the Asynchronous Request FIFO do not pass any posted writes, whether posted in the Physical or Asynchronous Request FIFOs. Within the Physical Request FIFO, read requests may coherently pass posted writes, but write requests and posted writes do not pass other writes posted in the Physical Request FIFO. Physical read and write requests may pass writes posted to the Asynchronous Request FIFO.

In conjunction with the ordering rules, the following protocol restrictions are adhered to so that proper ordering and therefore data integrity is maintained. The term "visible side-effect" is used to mean an indirect action caused by a request or response which results in the alteration of the contents or usage of host memory outside the address scope of the request or response.

1) Write requests within the range 0000_FFFF_FFFFh to FFFE_FFFF_FFFFh do not have 1394 visible side effects.
2) Read or write requests within the range 0h to 0_FFFF_FFFEh, whether handled by the Physical Request controller or not, do not have 1394 visible side-effects.
3) Read requests to CSR addresses which are processed autonomously by the CS4210 (Section 4.4.4 "Autonomous CSR Resources" on page 60) do not have 1394 visible side-effects.
4) If an error occurs in writing the posted data packet, the CS4210 sets an interrupt event to notify software and provides information about the failed write in an error reporting register. For more information about error handling of posted writes, refer to Section 3.7.7 "Posted Write Error" on page 29.

Operational Description (Continued)

3.5.4 Retries

For asynchronous receive, the CS4210 supports dual-phase retry for packets that must be busied. For asynchronous transmit, CS4210 supports the single-phase retry protocol. The retry mechanism is managed by hardware and invisible to software.

3.5.5 DMA Summary

Table 3-9 is a summary of DMA information for reference purposes.

Table 3-9. DMA Summary

DMA	Contexts	Per Context Registers	Per Context Interrupts	Receive Mode	DMA Commands	Z	tcodes
Asynchronous Transmit	1 Request	ContextControl CommandPtr	reqTxComplete		OUTPUT_MORE OUTPUT_MORE-Immediate OUTPUT_LAST OUTPUT_LAST-Immediate	2-8	0, 1, 4, 5, 9, A, E
	1 Response	ContextControl CommandPtr	respTxComplete				2, 6, 7, B
Asynchronous Receive	1 Request	ContextControl CommandPtr	ARRQ RQPkt	Buffer-fill	INPUT_MORE	1	0, 1, 4, 5, 9, E*
	1 Response	ContextControl CommandPtr	ARRS RSPkt				2, 6, 7, B
Isochronous Transmit	8	ContextControl CommandPtr	IsochTx IsochTxIntEventn IsochTxIntMaskn		OUTPUT_MORE OUTPUT_MORE-Immediate OUTPUT_LAST OUTPUT_LAST-Immediate STORE_VALUE	1-8	A
Isochronous Receive	8	ContextControl CommandPtr ContextMatch	IsochRx IsochRxIntEventn IsochRxIntMaskn	Packet-per-buffer	INPUT_MORE INPUT_LAST	1-8	A
				Buffer-fill	INPUT_MORE	1	
Self-ID	1	SelfIDBuffer SelfIDCount	SelfIDComplete	Buffer-fill		N/A	

Operational Description (Continued)

3.6 PHYSICAL REQUESTS

When a block or quadlet read or write request is received, the CS4210 handles the operation automatically without involving software if the offset address in the request packet header meets a specific set of criteria listed below. Requests that do not meet these criteria are directed to the AR DMA Request context unless otherwise specified. CS4210 registers which are written via physical access to the CS4210 yield unspecified results. The CS4210 checks to see if the offset address in the request packet header is one of the following.

If the offset falls within the physical range, then the offset address is used as the memory address for the block or quadlet transaction. Physical range is defined by offsets inclusively between a lower bound of 0h and an upper bound of 0000_FFFF_FFFFh. If the high order 16-bits of the offset address is 0000h, then the lower 32 bits of the offset address are used as the memory address for the block or quadlet transaction.

Lock transactions and block transactions with a non-zero extended tcode are not supported in this address space, instead they are diverted to the AR DMA Request context. For read requests, the information needed to formulate the response packet is passed to the Physical Response Unit. Requests are only accepted if the source node ID of the request has a corresponding bit in the Asynchronous Request Filter registers and Physical Request Filter registers (see Section 4.4.23 "Physical Request Filter Registers" on page 79).

If the offset address selects one of the following addresses, the physical request unit directly handles quadlet compare-swaps and quadlet reads. Other requests are sent an ack_type_error (see Table 3-7 on page 20.)

1) BUS_MANAGER_ID (FFFFF000021Ch):
   Local register is nscBusmgrID (BAR1+Offset 60h).
2) BANDWIDTH_AVAILABLE (FFFFF0000220h):
   Local register is nscBandwAvai (BAR1+Offset 64h).
3) CHANNELS_AVAILABLE_HI (FFFFF0000224h):
   Local register is nscChanAvailHi (BAR1+Offset 68h).
4) CHANNELS_AVAILABLE_LO (FFFFF0000228h):
   Local register is nscChanAvailLo (BAR1+Offset 6Ch).

If the offset address is one of the following addresses, the Physical Request controller directly handles quadlet reads. Other requests shall be sent an ack_type_error.

1) Config ROM header (1st quadlet of the Config ROM) (FFFFF0000400h): Local register is ConfigROM-header (see Section 4.4.5 "Configuration ROM Header Register" on page 61).
2) Bus ID (1st quadlet of the Bus_Info_Block) (FFFFF0000404h): Local register is BusID (see Section 4.4.6 "Bus Identification Register" on page 61).
3) Bus options (2nd quadlet of the Bus_Info_Block) (FFFFF0000408h): Local register is BusOptions (see Section 4.4.7 "Bus Options Register" on page 62).
4) Global unique ID (3rd and 4th quadlets of the Bus_Info_Block) (FFFFF000040Ch and FFFFF0000410h): Local registers are GlobalIDHi and GlobalIDLo (see Section 4.4.8 "Global Unique ID Register" on page 63).
5) Configuration ROM (FFFFF0000414h to FFFFF00007FFh). Mapped by the ConfigROMmapping register to a 1 KB block of system memory (see Section 4.4.9 "Configuration ROM Mapping Register" on page 63)

For information about ack codes for write requests, see Section 3.5.2 "Ack Codes for Write Requests" on page 24.

3.6.1 Filtering Physical Requests

Software can control which nodes it receives packets from by utilizing the asynchronous filter registers. There are two registers, one for filtering out all requests from a specified set of nodes (AsynchronousRequestFilter register) and one for filtering out physical requests from a specified set of nodes (PhysicalRequestFilter register). The settings in both registers have a direct impact on how the AR DMA Request context is used (e.g., disabling only physical receives from a node causes all request packets from that node to be routed to the AR DMA Request context). The usage and interrelationship between these registers is described in Section 4.4.22 "Asynchronous Request Filter Registers" on page 78."

Operational Description (Continued)

3.6.2 Posted Writes

For write requests handled by the Physical Request controller, the CS4210 may send an ack_complete before the data is actually written to system memory. These writes are referred to as posted writes since posted writes impact the Physical Request controller and the Asynchronous Receive Request DMA context. Further information about posted writes is located in Section 3.5.3 "Posted Writes" on page 24. Information on host bus error handling of posted writes is provided in Section 3.7.7 "Posted Write Error" on page 29."

3.6.3 Physical Responses

The response packet generated for a physical read, non-posted write, and lock request contains the transaction label as it appeared in the request, the destination_ID as provided in the request's source_ID, and are transmitted at the speed at which the request was received. The source bus ID in the response packet is equal to the destination bus ID from the original request. Note that this is not necessarily the same as the contents of the busNumber field in the Node ID register (BAR0+Offset E8h[15:6]). Unlike AR Response packets, physical responses do not track a SPLIT_TIMEOUT expiration time.

3.6.4 Physical Response Retries

There is a separate nibble-wide MaxPhysRespRetries field in the ATRetries Register (BAR0+Offset 08h[15:11]) that tells the Physical Response Unit how many times to attempt to retry the transmit operation for the response packet when an ack_busy* or ack_data_error is received from the target node. If the retry count expires, the packet is dropped and software is not notified. Refer to Section 4.4.3 "ATRetries Register" on page 59 for register details.

3.6.5 Interrupt Considerations for Physical Requests

Physical read request handling does not cause an interrupt to be generated under any circumstances. Physical write requests generate an interrupt when posted write processing yields an error. Lock requests to the serial bus registers generate an interrupt when the CS4210 is unable to deliver a lock response packet.

3.6.6 Bus Reset

On a bus reset, all pending physical requests (those for which ack_pending was sent) are discarded. Following a bus reset, only physical requests to the autonomous CSR resources (see Section 4.4.4 "Autonomous CSR Resources" on page 60) can be handled immediately. Other physical requests are processed after software initializes the filter registers (see Section 4.4.22 "Asynchronous Request Filter Registers" on page 78 and Section 4.4.23 "Physical Request Filter Registers" on page 79).

Operational Description (Continued)

3.7 HOST BUS ERRORS

The CS4210 has three primary goals when dealing with host bus error conditions:

1) Continue transmission and/or reception on all contexts not involved in the error.

2) Provide information to software which is sufficient to allow recovery from the error when possible.

3) Provide a means of error recovery on a context other than a general chip reset.

3.7.1 Causes of Host Bus Errors

Host bus errors can generally be classified as one of the following:

- Addressing error (e.g., non-existent memory location)
- Operation error (e.g., attempt to write to read-only memory)
- Data transfer error (e.g., parity or unrecoverable ECC)
- Time-out error (e.g., reply on split transaction bus was not received in time)

Each of these errors can occur at three identifiable stages in the processing of a descriptor:

- Descriptor fetch
- Data transfer (read or write)
- Optional descriptor status update.

In general, the nature of the bus error is not as significant as the stage of descriptor processing in which it occurs. For example, the difference between an addressing error and a data parity error is not significant to the error processing.

3.7.2 CS4210 Actions When Host Bus Error Occurs

When a host bus error occurs, the CS4210 performs a defined set of actions for all context types. Additionally, there is a set of actions that is performed depending on the context type. The following subsections outline these actions.

3.7.2.1 Descriptor Read Error

When an error occurs during the reading of a descriptor or descriptor block, the behavior of the CS4210 is the same regardless of the context type. The CS4210 sets ContextControl.dead and sets ContextControl.event to evt_descriptor_read to indicate that the descriptor fetch failed. The unrecoverable error IntEvent is generated and the context's IntEvent is not set. Additionally, CommandPtr is set to point to a descriptor within the descriptor block in which the error occurred. Since the descriptor could not be read, its xferStatus and resCount are not written with current values, and software must refer to ContextControl.*event* for the status.

3.7.2.2 xferStatus Write Error

For any type of context, when the CS4210 encounters an error writing the status to a descriptor, it sets ContextControl.dead. The values that would have been written to xferStatus of a descriptor are retained in ContextControl for inspection by system software. The unrecoverable error IntEvent is generated and the context's IntEvent is not set regardless of the setting of the interrupt (i) field in the descriptor. Additionally, CommandPtr is set to point to a descriptor within the descriptor block in which the error occurred.

3.7.2.3 Transmit Data Read Error

For asynchronous request transmit, asynchronous response transmit, and isochronous transmit, the CS4210 handles system data read errors in a similar manner. The CS4210 does not stop processing for the context. Instead, the event code in the status of the OUTPUT_LAST* descriptor is set to indicate that there was an error and the nature of the error. The indicated errors are evt_data_read or evt_underrun. If the error occurs before a packet's header is placed in the output FIFO, the CS4210 can immediately abort the packet transfer, optionally set the descriptor status to evt_data_read or evt_underrun, and move on to the next descriptor block. If the error occurs after the header has been placed in the output FIFO, the CS4210 stops placing data in the output FIFO. This causes the CS4210 to send a packet with a length that does not agree with the data_length field of the header. If the CS4210 receives an ack_data_error from the addressed node, then the CS4210 substitutes evt_data_read or evt_underrun as appropriate. If the device returns anything other than ack_data_error, then the CS4210 stores that value in the status for the packet. This means that if the addressed node returns an ack_pending on a block write, the error indication is lost.

If the packet was a broadcast write, an isochronous packet, or an asynchronous stream packet, no ack code is received from any node. In this case, the CS4210 assumes that ack_data_error was received and proceeds as outlined above.

Note: Underruns which occur due to host bus latency are not construed to be host bus data errors, and as a result such asynchronous request and response packets are retried as described in Section 4.4.3 "ATRetries Register" on page 59.

Operational Description (Continued)

3.7.3 Isochronous Transmit Data Write Error

A data write error can occur when the CS4210 attempts to write to the address indicated in a STORE_VALUE descriptor. This error is handled like a data read error with the exception that the event code is set to evt_data_write. The CS4210 does not begin placing the packet associated with a STORE_VALUE into the output FIFO until the STORE_VALUE operation is complete. This is to prevent the possibility of having multiple errors that cannot be properly reported to system software.

3.7.4 Asynchronous Receive DMA Data Write Error

When a host bus error occurs while the CS4210 is attempting to write to either the request or response buffer, the CS4210 sets the corresponding ContextControl.dead and set ContextControl.event to evt_data_write. The unrecoverable error IntEvent is generated and the context's IntEvent is not set regardless of the setting of the interrupt (i) field in the descriptor. CommandPtr.descriptorAddress points to the descriptor that contained the buffer descriptor for the memory address at which the error occurred. Any data in the input FIFO for the context is discarded.

3.7.5 Isochronous Receive Data Write Error

If a data write error occurs for a context that is in packet-per-buffer mode, the CS4210 sets ContextControl.*event* to evt_data_write or evt_overrun and conditionally updates xferStatus of the descriptor in which the error occurred. Any remaining data in the input FIFO for the packet is discarded. The resCount value in a descriptor that has an error will not necessarily reflect the correct number of data bytes successfully written to memory. If a FIFO overrun occurs for a context that is in buffer-fill mode, the packet is treated as if a data length error had occurred and is 'backed out' of the receive buffer (xferStatus and resCount not updated) and the remainder of the packet is discarded from the input FIFO. If a host bus error occurs for a context in buffer-fill mode, the CS4210 sets ContextControl.dead and sets ContextControl.event to evt_data_write. The unrecoverable error IntEvent is generated and the context's IntEvent is not set regardless of the setting of the interrupt (i) field in the descriptor. CommandPtr.descriptorAddress points to the descriptor that contained the buffer descriptor for the memory address at which the error occurred. Any data in the input FIFO for the context is discarded.

3.7.6 Physical Read Error

When an external node does a physical access and the CS4210's read of system memory fails, the CS4210 returns an error indication to the requester by forming a response containing a response code of resp_data_error. If the device replies with ack_busy or ack_data_error the host retries the packet. If the error was caused by a FIFO underrun, the CS4210 retries with the same response. If the error was a host bus error, the response packet is changed to resp_data_error.

3.7.7 Posted Write Error

Whether to be handled by the Physical Request controller or by the Asynchronous Receive Request context, write requests to certain address ranges (see Section 3.6 "Physical Requests" on page 26) may be acked with ack_complete before the data is actually written to system memory. Since the sending node has been notified that the action is complete, when the CS4210 cannot complete a posted write operation due to a host bus error the system must be notified so that software can recover.

If an error occurs in writing the posted data packet, then the CS4210 sets the IntEvent.PostedWriteErr bit (BAR0+Offset 80h[8]) to indicate that an error has occurred and the write remains pending. Software can then read the source node ID and offset address from the PostedWriteAddress register and then clear IntEvent.PostedWriteErr. When software clears IntEvent.PostedWriteErr, that write is no longer pending.

Although the CS4210 allows four pending writes, error reporting is through a single pair of software visible registers. If multiple posted write failures have occurred, software accesses them one at a time through the PostedWriteAddress register. When software clears IntEvent.PostedWriteErr, this is a signal to the CS4210 that software has completed reading of the current contents of PostedWriteAddress and that the CS4210 can report another error by again setting IntEvent.PostedWriteErr and presenting a new set of values when software reads PostedWriteAddress. Table 3-10 provides a map of the PostedWriteAddress register. Refer to Section 4.4.10 "PostedWriteAddress Register" on page 64 for further register information.

If the CS4210 has four pending physical writes, additional physical writes may not be posted. Instead the CS4210 returns ack_pending and only returns a complete indication when the write is actually done.

Table 3-10. BAR0+Offset 38h: PostedWriteAddress Register Map

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
sourceID																offsetHi															
offsetLo																															

Operational Description (Continued)

3.8 BUS RESETS

When a 1394 bus reset occurs, certain actions must be taken by software for proper operation of the DMA contexts. These actions and the behavior of the CS4210 are described in the following subsections.

3.8.1 Asynchronous Transmit

Upon detection of a bus reset, the CS4210 ceases transmission of asynchronous transmit packets. When this occurs there are two possibilities for AT packets that are left in the FIFO.

- Case 1 is when a bus reset occurs after the packet was transmitted but before an ack was received. For this category, the link side of the CS4210 returns evt_missing_ack.
- Case 2 is when a bus reset occurs after the packet is placed in the FIFO but before it is transmitted. For this category, the link side of the CS4210 returns evt_flushed.

When each context becomes stable (all data transfers have been halted and status writes have been completed), the CS4210 clears the corresponding ContextControl.active bit.

When a bus reset occurs, the link side flushes the asynchronous transmit FIFO(s) until the IntEvent.busReset condition is cleared. Software must make sure that IntEvent.busReset is not cleared until:

1) software has cleared the ContextControl.run bits for both Asynchronous Transmit contexts, and
2) both Asynchronous Transmit contexts have acquiesced and both ContextControl.active fields are zero. This is to ensure that all queued asynchronous packets (with potentially stale node numbers) are flushed.

Once the contexts are no longer active, software may clear the busReset interrupt condition, and hardware stops flushing the asynchronous transmit FIFO(s). Before setting ContextControl.run for either context following a bus reset, software must ensure that NodeID.iDValid is set and that NodeID.nodeNumber (Section 4.4.19 "Node ID and Status Register" on page 76) does not equal 63.

3.8.2 Asynchronous Receive

To assist software in determining which asynchronous request packets arrived before and after a bus reset, this is necessary since node numbers may have changed, the CS4210 inserts a synthesized CS4103 packet into the AR DMA Request Context buffer (if active) as soon as a bus reset condition is detected. The format of the packet can be found in the 1394 OHCI specification.

Software can distinguish the bus-reset packet from authentic CS4103 packets by the value of eventCode which is set to evt_bus_reset. Software can further interpret and coordinate received asynchronous packets across multiple bus resets by using the selfIDGeneration number provided in the bus-reset packet. Since the bus-reset packet is fabricated when a bus reset is initially detected, the selfIDGeneration number is for the new (not previous) generation and is the same as the selfIDGeneration number in the SelfID-Count register as well as in the selfID buffer. If more than one bus reset has occurred without any intervening packets, then only the “last” one is required to result in a synthesized bus-reset packet. If the input FIFO is full when a bus reset occurs, the link side of the FIFO inserts the bus-reset packet when space becomes available. If the AR DMA request context does not have enough buffer space for the bus-reset packet, the packet is synthesized once buffer space becomes available. The bus reset interrupt (IntEvent.busReset) is independent on the time when this packet goes from the FIFO into a host buffer. This interrupt shall occur as soon as possible after a bus reset has been detected. The bus-reset packet is no different from any other packet going into the AR Request buffer in that IntEvent.RQPkt is generated like it is for other packets.

3.8.3 Isochronous Transmit and Receive

Bus reset does not affect isochronous contexts.

Operational Description (Continued)

3.9 SERIAL EEPROM

A serial EEPROM may be used to configure the CS4210. If a serial EEPROM device is connected, the CS4210 detects the pull-up resistor on the EEDATA pin (pin 5) and considers the EEPROM present. Immediately after RST# is deasserted, the serial EEPROM is scanned via EEDATA and EECLK (pin 4). The CS4210 reads configuration information from the first 34 bytes of the EEPROM.

If the presence of the serial EEPROM was not detected, the CS4210 will not attempt to load configuration data via the EEDATA/EECLK pins. It will instead allow software to write to the guidHi and guidLo bit fields of the GUID register once after each hardware reset (RST#).

3.9.1 Serial EEPROM Cyclic Redundancy Check

The serial EEPROM uses a Cyclical Redundancy Check (CRC) to insure the data in the EEPROM is valid. If the CRC check fails, the data from the EEPROM is not loaded into the mapped registers listed in Table 3-11. The CRC1 and CRC2 values may be computed using the sample code in Figure 3-1 "CRC1 and CRC2 Sample Code" on page 32. This code accepts hexadecimal data from STDIN and writes data plus CRC to STDOUT. As a check, this code should produce the results as shown in Figure 3-1.

Table 3-11. Serial EEPROM Map

Offset	Description
00h	subsystem[7:0]
01h	subsystem[15:8]
02h	subsystem[23:16]
03h	subsystem[31:24]
04h	configROMheader[7:0]
05h	configROMheader[15:8]
06h	configROMheader[23:16]
07h	configROMheader[31:24]
08h	busOptions[7:0]
09h	busOptions[15:8]
0Ah	busOptions[23:16]
0Bh	busOptions[31:24]
0Ch	guidHi[7:0]
0Dh	guidHi[15:8]
0Eh	guidHi[23:16]
0Fh	guidHi[31:24]
10h	guidLo[7:0]
11h	guidLo[15:8]
12h	guidLo[23:16]
13h	guidLo[31:24]
14h	nscControl[7:0]
15h	nscControl[15:8]
16h	nscControl[23:16]
17h	nscControl[31:24]
18h	nscTxThrsh[7:0]
19h	nscTxThrsh[15:8]
1Ah	nscTxThrsh[23:16]
1Bh	nscTxThrsh[31:24]
1Ch	cmcControl[7:0]
1Dh	cmcControl[15:8]
1Eh	cmcControl[23:16]
1fh	cmcControl[31:24]
20h	CRC1
21h	CRC2

Operational Description (Continued)

```
ff ff ff ff ff ff ff ff -> CRC1 = feh, CRC2 = 70h
01 23 45 67 89 ab cd ef -> CRC1 = 19h, CRC2 = 07h
00 00 00 00 00 00 00 00 -> CRC1 = bfh, CRC2 = f4h
#include <stdio.h>
typedef unsigned char uchar_t;
typedef unsigned short ushort_t;
typedef unsigned long long_t;

ushort_t calc_crc(ushort_t r, uchar_t d);
ushort_t rev16(ushort_t src);
/* main routine */
int main(int argc, char *argv[])
{
   ushort_t residual = 0xffff;
   int din;
   uchar_t byte;
   uchar_t crc1, crc2;
   while(scanf("%2x", &din ) != EOF ) {
      byte = din;
      printf("%2.2x ", byte);
      residual = calc_crc(residual, byte);
   };
   printf("\n");
   residual = rev16(residual);
   crc1 = ~(residual & 0x00ff);
   crc2 =  ~(residual>>8);
   printf("CRC1 = %2.2hxh\n", crc1);
   printf("CRC2 = %2.2hxh\n", crc2);

   return(0);
/* calc_crc - apply a byte of data to the 16-bit CRC */
}

ushort_t calc_crc(ushort_t r, uchar_t d)
{
   ushort_t bit_in;
   int i;

   for(i=0;i<8;i++) {
      bit_in = (d>>i) & 0x0001;
      if((r>>15) ^ bit_in)
         r = (((r<<1) & 0x7ffb) | ((r>>15 ^ ((r>>14)&0x0001) ^ bit_in) << 15) |
             ((r>>15 ^ ((r>>1)&0x0001) ^ bit_in) << 2) | 0x0001 );
      else
         r <<= 1;
   }
   return(r);
}

ushort_t rev16(ushort_t src)
{
   return(
      ((src&0x0001)<<15) |
      ((src&0x0002)<<13) |
      ((src&0x0004)<<11) |
      ((src&0x0008)<<9) |
      ((src&0x0010)<<7) |
      ((src&0x0020)<<5) |
      ((src&0x0040)<<3) |
      ((src&0x0080)<<1) |
      ((src&0x0100)>>1) |
      ((src&0x0200)>>3) |
      ((src&0x0400)>>5) |
      ((src&0x0800)>>7) |
      ((src&0x1000)>>9) |
      ((src&0x2000)>>11) |
      ((src&0x4000)>>13) |
      ((src&0x8000)>>15);
}
```

Figure 3-1. CRC1 and CRC2 Sample Code

4.0 Register Descriptions

The registers of the CS4210 can broadly be divided into two categories:

1) OHCI Configuration Registers

2) National (Vendor) Specific Configuration Registers

Both are memory mapped offsets accessed via Base Address Registers (BARs) specified in the PCI Configuration Space Header.

The remaining sub-sections of this chapter are as follows:

- A brief discussion on how to access the registers located in the PCI Configuration Space
- Register Summary
- Detailed bit formats of all registers

4.1 PCI CONFIGURATION SPACE ACCESS

A PCI configuration read or write cycle is accomplished by writing the bus, function, device, and register number into the 32-bit index register at 0CF8h, then performing a corresponding PCI configuration read or write of the 32-bit data register at 0CFCh. The format of the value written to 0CF8h (PCI_INDEX) is shown in Table 4-1.

The CS4210 provides a configuration space, whose first 40h bytes adhere to the format outlined in Revision 2.1 of the document entitled, “PCI Local Bus Specification”. This is a Type 0 header. This configuration space header contains two 32-bit BARs that specify memory space usage. The first BAR contains the memory address of the OHCI defined registers. The second BAR contains the memory address of the National Semiconductor defined registers.

Table 4-1. PCI Index Register (0CF8h)

31	30 24	23 16	15 11	10 8	7 2	1 0
Configuration Space Mapping	**RSVD**	**Bus Number**	**Device Number**	**Function Number**	**Register Number**	**00**
1 (Enable)	000 000	0000 0000	xxxx x	xxx	xxxx xx	00 (Always)

Register Descriptions (Continued)

4.2 REGISTER SUMMARY

The tables in this subsection summarize all the registers of the CS4210. Included in the tables are the register's reset values and page references where the bit formats are found.

Table 4-2. PCI Configuration Registers Summary

Index	Width (Bits)	Access	Name	Reset Value	Reference (Page)
00h-01h	16	R	Vendor Identification Register	1000h	Page 42
02h-03h	16	R	Device Identification Register	000Fh	Page 42
04h-05h	16	RW	PCI Command Register	0000h	Page 42
06h-07h	16	RW	PCI Status Register	0200h	Page 43
08h	8	R	Device Revision ID Register	03h	Page 43
09h-0Bh	24	R	PCI Class Code Register	0C0010h	Page 43
0Ch	8	R	PCI Cache Line Size Register	00h	Page 43
0Dh	8	RW	PCI Latency Timer Register	50h	Page 43
0Eh	8	R	PCI Header Type Register	00h	Page 43
0Fh	8	R	PCI BIST Register	00h	Page 43
10h-13h	32	RW	Base Address Register 0 (BAR0): Sets base address for memory mapped OHCI Configuration Registers	00000000h	Page 44
14h-17h	32	RW	Base Address Register 1 (BAR1): Sets base address for memory mapped National Semiconductor device specific operational registers.	00000000h	Page 44
18h-2B	--	--	Reserved	---	Page 44
2Ch-2Dh	16	R	Subsystem Vendor Identification Register	1	Page 44
2Eh-2Fh	16	R	Subsystem Identification Register	2	Page 44
30h-33h	--	--	Reserved	---	Page 44
34h	8	R	Capabilities Pointer Register	3	Page 45
35h-3Bh	--	--	Reserved	---	Page 45
3Ch	8	RW	Interrupt Line Register	FFh	Page 45
3Dh	8	R	Interrupt Pin Register	01h	Page 45
3Eh	8	R	Min Grant Register	00h	Page 45
3Fh	8	R	Max Latency Register	00h	Page 45
40h-43h	32	RW	PCI HCI Control Register	00h	Page 45
44h	8	R	Capability ID Register	01h	Page 45
45h	8	R	Next Item Pointer Register	00h	Page 45
46h-47h	16	R	Power Management Capabilities Register	4	Page 46
48h-49h	16	R/W	Power Management and Control Status Register	5	Page 46
4Ah	8	R	Power Management CSR Bridge Support Extension Register	00h	Page 47
4Bh	8	R	Power Management Data Register	00h	Page 47

1. The reset value must be set in the serial EEPROM to the vendor identification number assigned by the PCI SIG.
2. The reset value must be set in serial EEPROM to a unique number chosen by the user to represent this PCI device implementation.
3. The reset value is dependent upon the PCICapabilities bit in the nscControl register (BAR1+Offset 00h[20]) which can be configured by the serial EEPROM. If enabled in the EEPROM, the reset value of the register is 44h. If not enabled in the EEPROM or if no EEPROM is present, then the reset value is 00h.
4. The reset value is dependent upon the PCICapabilities bit in the nscControl register (BAR1+Offset 00h[20]) which can be configured by serial EEPROM. If enabled in the EEPROM the reset value of this register is 4000h. If not or if no EEPROM is present, then the reset value is 000h.
5. The reset value is dependent upon the PCICapabilities bit in the nscControl register (BAR1+Offset 00h[20]) which can be configured by serial EEPROM. If enabled in the EEPROM the reset value of this register is 8000h. If not or if no EEPROM is present then the reset value is 000h.

Register Descriptions (Continued)

Table 4-3. OHCI Configuration Registers Summary

BAR0+Offset	Width (Bits)	Access	Name	Reset Value	Reference (Page)
00h-03h	32	R	Version Register	xxh	Page 58
04h-07h	32	RSU/RU	GUIDROM Register	xxh	Page 58
08h-0Bh	32	RW	ATRetries Register	xxh	Page 59
0Ch-0Fh	32	RWU	CSRReadData Register	xxh	Page 60
10h-13h	32	RW	CSRCompareData Register	xxh	
14h-17h	32	RW	CSRControl Register	800xh	
18h-1Bh	32	RWU	Configuration ROM Header Register	00h	Page 61
1Ch-1Fh	32	R	Bus Identification Register	31333934h	Page 61
20h-23h	32	RW	Bus Options Register	xxh	Page 62
24h-27h	32	RW	GUIDHi Register	xxh	Page 63
28h-2Bh	32	R/W	GUIDLo Register	xxh	
2Ch-33h	--	--	Reserved	00h	--
34h-37h	32	RW	Configuration ROM Map Register	xxh	Page 63
38h-3Bh	32	RU	PostedWriteAddressLo Register	xxh	Page 64
3Ch-3Fh	32	RU	PostedWriteAddressHi Register	xxh	
40h-43h	32	R	Vendor ID Register	80017h	Page 64
50h-53h	32	RSC	HCControl Set Register	xxh	Page 65
54h-57h	32	RSC	HCControl Clear Register	xxh	
58h-63h	--	--	Reserved	00h	--
64h-67h	32	RW	Self-ID Buffer Pointer Register	xxh	Page 68
68h-6Bh	32	RU	Self-ID Count Register	xxh	Page 68
70h-73h	32	RSC	IRMultiChanMaskHi Set Register	xxh	Page 69
74h-77h	32	RSC	IRMultiChanMaskHi Clear Register	xxh	
78h-7Bh	32	RSC	IRMultiChanMaskLo Set Register	xxh	
7Ch-7Fh	32	RSC	IRMultiChanMaskLo Clear Register	xxh	
80h-83h	32	RSCU	IntEvent Set Register	xxh	Page 70
84h-87h	32	RSCU	IntEvent Clear Register	xxh	
88h-8Bh	32	RSCU	IntMask Set Register	xxh	Page 72
8Ch-8Fh	32	RSCU	IntMask Clear Register	xxh	
90h-93h	32	RSC	IsochTxIntEvent Set Register	xxh	Page 73
94h-97h	32	RSC	IsochTxIntEvent Clear Register	xxh	
98h-9Bh	32	RSC	IsochTxIntMask Set Register	xxh	Page 73
9Ch-9Fh	32	RSC	IsochTxIntMask Clear Register	xxh	
A0h-A3h	32	RSC	IsochRxIntEvent Set Register	xxh	Page 74
A4h-A7h	32	RSC	IsochRxIntEvent Clear Register	xxh	
A8h-ABh	32	RSC	IsochRxIntMask Set Register	xxh	Page 74
ACh-AFh	32	RSC	IsochRxIntMask Clear Register	xxh	
B0h-DBh	--	--	Reserved	00h	--
DCh-DFh	32	RW	Fairness Control Register	xxh	Page 75
E0h-E3h	32	RSC	LinkControl Set Register	xxh	Page 75
E4h-E7h	32	RSC	LinkControl Clear Register	xxh	
E8h-EBh	32	RU	Node ID and Status Register	xxh	Page 76
ECh-EFh	32	RWU	PHYControl Register	xxh	Page 77
F0h-F3h	32	RWU	IsochCycleTimer Register	xxh	Page 77
F4h-FFh	--	--	Reserved	00h	--
100h-103h	32	RSCU	AsyncRequestFilterHi Set Register	00h	Page 78
104h-107h	32	RSCU	AsyncRequestFilterHi Clear Register	00h	

Register Descriptions (Continued)

Table 4-3. OHCI Configuration Registers Summary (Continued)

BAR0+Offset	Width (Bits)	Access	Name	Reset Value	Reference (Page)
108h-10Bh	32	RSCU	AsyncRequestFilterLo Set Register	00h	Page 78
10Ch-10Fh	32	RSCU	AsyncRequestFilterLo Clear Register	00h	
110h-113h	32	RSCU	PhysicalRequestHi Set Register	00h	Page 79
114h-117h	32	RSCU	PhysicalRequestHi Clear Register	00h	
118h-1Bh	32	RSCU	PhysicalRequestLo Set Register	00h	Page 79
11Ch-11Fh	32	RSCU	PhysicalRequestLo Clear Register	00h	
180h-183h	32	RSU	AsyncReqTxContextControl Set Register	xxh	Page 80
184h-187h	32	RSU	AsyncReqTxContextControl Clear Register	xxh	
188h-18Bh	--	--	Reserved	00h	--
18Ch-18Fh	32		AsyncReqTxCommandPtr Register	xxh	Page 80
190h-19Fh	--	--	Reserved	00h	--
1A0h-1A3h	32	RSCU/RU	AsyncRespTxContextControl Set Register	xxh	Page 81
1A4h-1A7h	32	RSCU/RU	AsyncRespTxContextControl Clear Register	xxh	
1ACh-1AFh	32	RWU	AsyncRespTxCommandPtr Register	xxh	Page 81
1B0h-1BFh	--	--	Reserved	00h	--
1C0h-1C3h	32	RSCU/RSU	AsyncReqRxContextControl Set Register	xxh	Page 82
1C4h-1C7h	32		AsyncReqRxContextControl Clear Register	xxh	
1C8h-1CBh	--	--	Reserved	00h	--
1CCh-1CFh	32		AsyncReqRxCommandPtr Register	xxh	Page 82
1D0h-1DFh	--	--	Reserved	00h	--
1E0h-1E3h	32	RSCU/RU	AsyncRespRxContextControl Set Register	xxh	Page 83
1E4h-1E7h	32	RSCU/RU	AsyncRespRxContextControl Clear Register	xxh	
1E8h-1EBh	--	--	Reserved	00h	--
1ECh-1EFh	32	RWU	AsyncRespRxCommandPtr Register	xxh	Page 83
200h-203h	32	RSCU/RSU	IsochTx0ContextControl Set Register	xxh	Page 85
204h-207h	32	RSCU/RSU	IsochTx0ContextControl Clear Register	xxh	
208h-20Bh	--	--	Reserved	00h	--
20Ch-20Fh	32	RWU	IsochTx0CommandPtr Register	xxh	Page 85
210h-213h	32	RSCU/RSU	IsochTx1ContextControl Set Register	xxh	Page 85
214h-217h		RSCU/RSU	IsochTx1ContextControl Clear Register	xxh	
218h-21Bh	--	--	Reserved	00h	--
21Ch-21Fh	32	RWU	IsochTx1CommandPtr Register	xxh	Page 85
220h-223h	32	RSCU/RSU	IsochTx2ContextControl Set Register	xxh	Page 85
224h-227h	32	RSCU/RSU	IsochTx2ContextControl Clear Register	xxh	
228h-22Bh	--	--	Reserved	00h	--
22Ch-22Fh	32	RWU	IsochTx2CommandPtr Register	xxh	Page 85
230h-233h	32	RSCU/RSU	IsochTx3ContextControl Set Register	xxh	Page 85
234h-237h	32	RSCU/RSU	IsochTx3ContextControl Clear Register	xxh	
238h-23Bh	--	--	Reserved	00h	--
23Ch-23Fh	32	RWU	IsochTx3CommandPtr Register	xxh	Page 85
240h-243h	32	RSCU/RSU	IsochTx4ContextControl Set Register	xxh	Page 85
244h-247h	32	RSCU/RSU	IsochTx4ContextControl Clear Register	xxh	
248h-24Bh	--	--	Reserved	00h	--
24Ch-24Fh	32	RWU	IsochTx4CommandPtr Register	xxh	Page 85
250h-253h	32	RSCU/RSU	IsochTx5ContextControl Set Register	xxh	Page 85
254h-257h	32	RSCU/RSU	IsochTx5ContextControl Clear Register	xxh	
258h-25Bh	--	--	Reserved	00h	--
25Ch-25Fh	32	RWU	IsochTx5CommandPtr Register	xxh	Page 85

Register Descriptions (Continued)

Table 4-3. OHCI Configuration Registers Summary (Continued)

BAR0+Offset	Width (Bits)	Access	Name	Reset Value	Reference (Page)
260h-263h	32	RSCU/RSU	IsochTx6ContextControl Set Register	xxh	Page 85
264h-267h	32	RSCU/RSU	IsochTx6ContextControl Clear Register	xxh	
268h-26Bh	--	--	Reserved	00h	--
26Ch-26Fh	32	RWU	IsochTx6CommandPtr Register	xxh	Page 85
270h-273h	32	RSCU/RSU	IsochTx7ContextControl Set Register	xxh	Page 85
274h-277h	32	RSCU/RSU	IsochTx7ContextControl Clear Register	xxh	
278h-27Bh	--	--	Reserved	00h	--
27Ch-27Fh	32	RWU	IsochTx7CommandPtr Register	xxh	Page 85
280h-3FFh	--	--	Reserved	00h	--
400h-403h	32	RSCU/RSU	IsochRx0ContextControl Set Register	xxh	Page 87
404h-407h	32	RSCU/RSU	IsochRx0ContextControl Clear Register	xxh	
408h-40Bh	--	--	Reserved	00h	--
40Ch-40Fh	32	RWU	IsochRx0CommandPtr Register	xxh	Page 88
410h-413h	32	RW	IsochRx0ContextMatch	xxh	Page 88
414h-41Fh	--	--	Reserved	00h	--
420h-423h	32	RSCU/RSU	IsochRx1ContextControl Set Register	xxh	Page 87
424h-427h	32	RSCU/RSU	IsochRx1ContextControl Clear Register	xxh	
428h-42Bh	--	--	Reserved	00h	--
42Ch-42Fh	32	RWU	IsochRx1CommandPtr Register	xxh	Page 88
430h-433h	32	RW	IsochRx1ContextMatch Register	xxh	Page 88
434h-43Fh	--	--	Reserved	00h	--
440h-443h	32	RSCU/RSU	IsochRx2ContextControl Set Register	xxh	Page 87
444h-447h	32	RSCU/RSU	IsochRx2ContextControl Clear Register	xxh	
448h-44Bh	--	--	Reserved	00h	--
44Ch-44Fh	32	RWU	IsochRx2CommandPtr Register	xxh	Page 88
450h-453h	32	RW	IsochRx2ContextMatch Register	xxh	Page 88
454h-45Fh	--	--	Reserved	00h	--
460h-463h	32	RSCU/RSU	IsochRx3ContextControl Set Register	xxh	Page 87
464h-467h	32	RSCU/RSU	IsochRx3ContextControl Clear Register	xxh	
468h-46Bh	--	--	Reserved	00h	--
46Ch-46Fh	32	RWU	IsochRx3CommandPtr Register	xxh	Page 88
470h-473h	32	RW	IsochRx3ContextMatch Register	xxh	Page 88
474h-47Fh	--	--	Reserved	00h	--
480h-483h	32	RSCU/RSU	IsochRx4ContextControl Set Register	xxh	Page 87
484h-487h	32	RSCU/RSU	IsochRx4ContextControl Clear Register	xxh	
488h-48Bh	--	--	Reserved	00h	--
48Ch-48Fh	32	RWU	IsochRx4CommandPtr Register	xxh	Page 88
490h-493h	32	RW	IsochRx4ContextMatch Register	xxh	Page 88
494h-49Fh	--	--	Reserved	00h	--
4A0h-403h	32	RSCU/RSU	IsochRx5ContextControl Set Register	xxh	Page 87
4A4h-407h	32	RSCU/RSU	IsochRx5ContextControl Clear Register	xxh	
4A8h-40Bh	--	--	Reserved	00h	--
4ACh-40Fh	32	RWU	IsochRx5CommandPtr Register	xxh	Page 88
4B0h-413h	32	RW	IsochRx5ContextMatch Register	xxh	Page 88
4B4h-41Fh	--	--	Reserved	00h	--
4C0h-4C3h	32	RSCU/RSU	IsochRx6ContextControl Set Register	xxh	Page 87
4C4h-4C7h	32	RSCU/RSU	IsochRx6ContextControl Clear Register	xxh	
4C8h-4CBh	--	--	Reserved	00h	--

Register Descriptions (Continued)

Table 4-3. OHCI Configuration Registers Summary (Continued)

BAR0+Offset	Width (Bits)	Access	Name	Reset Value	Reference (Page)
4CCh-4CFh	32	RWU	IsochRx6CommandPtr Register	xxh	Page 88
4D0h-4D3h	32	RW	IsochRx6ContextMatch Register	xxh	Page 88
4D4h-4DFh	--	--	Reserved	00h	--
4E0h-4E3h	32	RSCU/RSU	IsochRx7ContextControl Set Register	xxh	Page 87
4E4h-4E7h	32	RSCU/RSU	IsochRx7ContextControl Clear Register	xxh	
4E8h-4EBh	--	--	Reserved	00h	--
4ECh-4EFh	32	RWU	IsochRx7CommandPtr Register	xxh	Page 88
4F0h-4F3h	32	RW	IsochRx7ContextMatch Register	xxh	Page 88
4F4h-4FFh	--	--	Reserved	00h	--

Register Descriptions (Continued)

Table 4-4. National Specific Configuration Registers Summary

BAR1+Offset	Width (Bits)	Access	Name	Reset Value	Reference (Page)
00h-03h	32	RW	nscControl Register	00h	Page 91
04h-07h	32	RSCU	nscEvent Set Register	xxh	Page 93
08h-0Bh	32	RSCU	nscEvent Clear Register	xxh	
0Ch-0Fh	32	RSC	nscEventMask Set Register	xxh	Page 93
10h-13h	32	RSC	nscEventMask Clear Register	xxh	
14h-17h	32	RU/RW	nscRAMBist Register	xxh	Page 94
18h-1Bh	32	RW	nscCmcControl Register	00h	Page 94
20h-23h	32	RW/RU	nscTxThreshold Register	01FE01FEh	Page 94
24h-27h	32	RW	nscSubSystem Register	00h	Page 94
40h-43h	32	RW	nscPhysReadCount Register	00h	Page 95
44h-47h	32	RW	nscPhysWriteCount Register	00h	Page 95
48h-4Bh	32	RW	nscPhysLockCount Register	00h	Page 95
4Ch-5Fh	--	--	Reserved	00h	--
60h-63h	32	R	nscBusmgrID Register	xxh	Page 96
64h-67h	32	R	nscBandwAvail Register	xxh	Page 96
68h-6Bh	32	R	nscChanAvailHi Register	xxh	Page 96
6Ch-6Fh	32	R	nscChanAvailLo Register	xxh	Page 96

Register Descriptions (Continued)

4.3 PCI CONFIGURATION REGISTERS

The PCI configuration space for the CS4210 is header type 0. Header type 0 is the format for the device's configuration header region which is the first 16 DWORDs of PCI configuration space. The configuration and operational registers are memory mapped into PCI memory address space and pointed to by Base Address Registers (BARs) in the PCI configuration space. PCI configuration space is not directly memory or I/O mapped - its access is system dependent. A software reset issued through the HCControl register does not affect the contents of the PCI configuration space.

Table 4-5 is a map for the PCI Configuration Registers. Table 4-6 gives detailed bit information.

Table 4-5. PCI Configuration Register Map: Index xxh

Index	Bits 31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h-03h	**Device Identification Register**																**Vendor Identification Register**															
	Device ID																Vendor ID															
04h-07h	**Status Register**																**Command Register**															
	detectedParErr	signalSysErr	recvMasterAbort	recvTargeAbort	signalTargetAbort	devsel Timing		dataParityRep	RSVD								RSVD						fastBBEn	systemErrEn	waitCycEn	parityErrResp	VGAPalSnoop	memWrInvalid	specCycRec	masterEn	memoryAccess	ioAccess
08h-0Bh	**PCI Device Class Code**																								**Revision ID**							
	Base Class								Sub Class								Programming Interface								revisionID							
0Ch-0Fh	**PCI BIST Register**								**PCI Header Type**								**PCI Latency Timer Register**								**PCI Cache Line Size Register**							
	BIST								Header Type								latencyTimer								cacheLineSize							
10h-13h	**Base Address Register 0 - OHCI Configuration Registers**																															
	Base Address 0																RSVD													Prefetchable	TP	IND
14h-17h	**Base Address Register 1 - National Specific Configuration Registers**																															
	Base Address 0																RSVD													Prefetchable	TP	IND
18h-2Bh	**Reserved**																															
2Ch-2Fh	**Subsystem ID**																**Subsystem Vendor ID**															
	Subsystem ID																Subsys Vend ID															
30h-33h	**Reserved**																															
34h-37h	**Reserved**																								**Capabilities**							
	RSVD																								Capabilities Pointer							
38h-3Bh	**Reserved**																															
3Ch-3Fh	**Max Latency**								**Min Grant**								**Interrupt Pin**								**Interrupt Line**							
	Max Latency								Min Grant								Interrupt Pin								Interrupt Line							
40h-43h	**Reserved**																															

Register Descriptions (Continued)

Table 4-5. PCI Configuration Register Map: Index xxh (Continued)

Index	Bits																													
	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
44h-47h	**Power Management Capabilities**																**Next Item Pointer**								**Capability ID**							
	pmeSupport					d2Support	daSupport	dynData	RSVD		DSI	auxPower	pmeCLK	pmVersion			NextItemPointer								CapabilityID							
48h-4Bh	**Power Management Data**								**Power Management CSR Bride Support Extensions**								**Power Management Control and Status**															
	pmData								pmcrBSE								pmeStatus	dynData						pmeEnab	RSVD			dynDataEnab	RSVD			pmeState

Register Descriptions (Continued)

Table 4-6. PCI Configuration Register Definitions

Bit	Name	Access	Reset	Description
Index 00h			**Vendor Identification Register**	
15:0	Vendor ID	RO	1000h	**Vendor Identification:** This register identifies the manufacturer of the CS4210 as National Semiconductor.
Index 02h			**Device Identification Register**	
15:0	Device ID	RO	000Fh	**Device Identification:** This register identifies the CS4210 as the IEEE 1394 Open Host Controller.
Index 04h			**PCI Command Register**	
This register provides coarse control over the device's ability to generate and respond to PCI cycles. It is required that the CS4210 support both PCI bus-mastering and memory-mapping of all configuration and operational registers into the memory address space of the PC host. Consequently, the fields memoryAccess and masterEn should always be set to 1 during device configuration. Once the CS4210 starts processing DMA descriptor lists, the action of resetting either field memoryAccess or masterEn to 0 halts all PCI operations. If the field memoryAccess is reset to 0, the CS4210 can no longer respond to any software command addressed to it and interrupt generation is halted.				
15:10	RSVD	---	000000	**Reserved**
9	fastBBEn	RO	0	**Fast Back-to-Back Enable**: This function is not supported and is always disabled. 0 = Disable; 1 = Enable.
8	systemErrEn	R/W	0	**System Error Enable:** Allow assertion on detection of special errors. 0 = Disable; 1 = Enable.
7	waitCycEn	RO	0	**Wait Cycle Control:** This function is not supported and is always disabled. 0 = Disable; 1 = Enable.
6	parityErrResp	R/W	0	**Parity Error Response:** Allow the CS4210 to drive PERR# when a parity error is detected. 0 = Disable; 1 = Enable.
5	VGAPalSnoop	RO	0	**VGA Palette Snoop:** This function is not supported, Is is always disabled. 0 = Disable; 1 = Enable.
4	memWrInvalid	RO	0	**Memory Write and Invalidate:** Allow the CS4210 to do memory write and invalidate cycles. 0 = Disable; 1 = Enable. If disabled, memory write commands must be used.
3	specCycRec	RO	0	**Special Cycles:** This function is not supported and is always disabled. 0 = Disable; 1 = Enable.
2	masterEn	R/W	0	**Bus Master:** Allow the CS4210 bus mastering capabilities. 0 = Disable; 1 = Enable.
1	memoryAccess	R/W	0	**Memory Space Access:** Allow the CS4210 to respond to memory cycles from the PCI bus. 0 = Disable; 1 = Enable. This bit must be set to 1 to access memory offsets through BAR0 (Index 10h) and BAR1 (Index 14h).
0	ioAccess	RO	0	**I/O Space Access:** Allow the CS4210 to respond to I/O cycles from the PCI bus. 0 = Disable; 1 = Enable.

Register Descriptions (Continued)

Table 4-6. PCI Configuration Register Definitions

Bit	Name	Access	Reset	Description
Index 06h				**PCI Status Register**
15	detectedParErr	R/W	0	**Detected Parity Error:** This bit is set whenever a parity error is detected. Write 1 to clear.
14	signalSysErr	R/W	0	**Signaled System Error:** This bit is set whenever the CS4210 asserts SERR# active. Write 1 to clear.
13	recvMasterAbort	R/W	0	**Received Master Abort:** This bit is set whenever a master abort cycle occurs while the CS4210 is the master for the PCI cycle. Write 1 to clear.
12	recvTargeAbort	R/W	0	**Received Target Abort:** This bit is set whenever a target abort is received while the CS4210 is the master for the PCI cycle. Write 1 to clear.
11	signalTargetAbort	R/W	0	**Signaled Target Abort:** This bit is set whenever the CS4210 signals a target abort while it is the target for the PCI cycle. Write 1 to clear.
10:9	devselTiming	RO	01	**DEVSEL# Timing:** These bits are always 01, as the CS4210 always responds to cycles for which it is an active target with medium DEVSEL# timing: 00 = Fast; 01 = Medium; 10 = Slow; 11 = Reserved
8	dataParityRep	R/W	0	**Data Parity Detected:** This bit is set when the CS4210 asserted PERR# or observed PERR# asserted. The parityErrResp in the Command Register (Index 04h[6]) must be enabled for this bit to function. Write 1 to clear.
7:0	RSVD	RO	0	**Reserved**
Index 08h				**Revision Identification Register**
7:0	revisionID	RO	03h	**Revision Identification:** Specifies the silicon revision as 03h. This value will be incremented for subsequent revisions.
Index 09h				**PCI Class Code Register**
23:16	Base Class	RO	0Ch	**Base Class:** Identifies the device as being a serial bus controller.
15:8	Sub Class	RO	00h	**Sub Class:** Identifies the device as being of IEEE 1394 class.
7:0	Programming Interface	RO	10h	**Programming Interface:** Identifies the device as being a 1394 OpenHCI controller.
Index 0Ch				**PCI Cache Line Size Register**
7:0	cacheLineSize	RO	00h	**PCI Cache Line Size:** This register sets the size of the PCI cache line. A value of 00h indicates caching is disabled.
Index 0Dh				**PCI Latency Timer Register**
7:0	latencyTimer	R/W	50h	**PCI Latency Timer Value:** This register contains the maximum number of PCI clocks that the CS4210 can hold ownership of the PCI bus as the bus master. Bits [3:0] are a constant 0.
Index 0Eh				**PCI Header Type Register**
7:0	Header Type	RO	00h	**PCI Header Type Register:** This register defines the format of this header. This header is of type format 0. Additionally, bit 7 defines whether this PCI device is a multi-function (bit 7 = 1) or single-function (bit 7 = 0) device.
Index 0Fh				**PCI BIST Register**
7:0	BIST	RO	00h	**PCI Built-In Self-Test:** A value of 00h indicates no PCI controlled BIST.

Register Descriptions (Continued)

Table 4-6. PCI Configuration Register Definitions

Bit	Name	Access	Reset	Description
Index 10h				**Base Address Register 0 (BAR0)**
This register specifies the base address of a contiguous memory space in the PCI memory space of the host. This memory space is assigned to the configuration and operational registers defined by the OHCI specification. The registers designated as the CS4210 Configuration Registers (listed in Section 4.4 "OHCI Configuration Registers" on page 48) are directly mapped into the first 2 KB of this memory space.				
32:11	Base Address 0	R/W	00h	**Base Memory Address:** For accessing the configuration registers defined by the CS4210. A 2 KB address range is used.
10:4	RSVD	--	0	**Reserved**
3	Prefetchable	RO	0	**Pre-fetchable:** 0 indicates the memory is not pre-fetchable.
2:1	TP	RO	00	**Target Pointer:** 00 indicates the base register is 32 bits wide and can be placed anywhere in the 32-bit memory space.
0	IND	RO	0	**Index:** 0 indicates the CS4210 configuration registers are mapped into memory space of the host system.
Index 14h				**Base Address Register 1 (BAR1)**
This register specifies the base address of a contiguous memory space in the PCI memory space of the host. This memory space is assigned to the operational registers defined by National (listed in Section 4.5 "National (NSC) Specific Configuration Registers" on page 89). These registers are mapped into the first 2 KB of this memory space.				
32:11	Base Address 0	R/W	00h	**Base Memory Address:** For accessing the vendor defined registers in the CS4210. A 2 KB address range is used.
10:4	RSVD	--	0	**Reserved**
3	Prefetchable	RO	0	**Pre-fetchable:** 0 indicates the memory is not pre-fetchable.
2:1	TP	RO	00	**Target Pointer:** 00 indicates the base register is 32 bits wide and can be placed anywhere in the 32-bit memory space.
0	IND	RO	0	**Index:** 0 indicates the CS4210 configuration registers are mapped into memory space of the host system.
Index 18h-2Bh				**Reserved**
Index 2Ch				**Subsystem Vendor Identification Register**
15:0	Subsys Vend ID	RO	See Note	**Subsystem Vendor Identification:** This register identifies the vendor of the subsystem that contains this OpenHCI function. The ID is assigned by the PCI SIG and is loaded from the serial ROM after power-up reset. This register can be accessed through NSC register space but cannot be written from PCI.
Note: The reset value must be set in the serial EEPROM to the vendor identification number assigned by the PCI SIG.				
Index 2Eh				**Subsystem Identification Register**
15:0	Subsys ID	RO	See Note	**Subsystem Identification:** This register identifies the subsystem that contains this OpenHCI function. The ID is assigned by the vendor and is loaded from the serial ROM after power-up reset. This register can be accessed through NSC register space but cannot be written from PCI
Note: The reset value must be set in serial EEPROM to a unique number chosen by the user to represent this PCI device implementation.				
Index 30h-33h				**Reserved**

Register Descriptions (Continued)

Table 4-6. PCI Configuration Register Definitions

Bit	Name	Access	Reset	Description
Index 34h				**Capabilities Pointer Register**
7:0	Capabilities Pointer	RO	See Note	**Capabilities Pointer**: This register provides a pointer into the PCI configuration header where the PCI power management register block resides. The configuration header registers residing at Index 44h and 48h provide the power management registers. The presence of this register is controlled by the PCICapabilities bit in the nscControl register (BAR1+Offset 00h[20]). If enabled this register points to Index 44h otherwise it reads zero.
Note: The reset value is dependent upon the PCICapabilities bit in the nscControl register (BAR1+Offset 00h[20]) which can be configured by the serial EEPROM. If enabled in the EEPROM, the reset value of the register is 44h. If not enabled in the EEPROM or if no EEPROM is present, then the reset value is 00h.				
Index 35h-3Bh				**Reserved**
Index 3Ch				**Interrupt Line Register**
7:0	Interrupt Line	R/W	FFh	**Interrupt Line:** This register is used to identify which of the system interrupt lines on the interrupt controller the CS4210 interrupt pin is routed to. The reset value of FFh indicates no connection.
Index 3Dh				**Interrupt Pin Register**
7:0	Interrupt Pin	RO	01h	**Interrupt Pin:** This register defines which of the four PCI interrupt request pins this device uses. The CS4210 uses INTA#.
Index 3Eh				**Min Grant Register**
7:0	Min Grant	RO	00h	**Min Grant:** This register specifies how many 250 ns periods are required by the CS4210 for burst transfers. A reset value of 00h specifies no stringent requirements on burst lengths.
Index 3Fh				**Max Latency Register**
7:0	Max Lat	RO	00h	**Max Latency:** This register defines how quickly the CS4210 requires the PCI bus after its REQ# has been asserted. The value of zero indicates there are no stringent requirements for PCI bus latency.
Index 40h-43h				**Reserved**
Index 44h				**Capability ID Register**
7:0	CapabilityID	RO	01h	**Capability ID:** This register specifies that the CS4210 supports PCI power management. It reads zero if the Capabilities Pointer (Index 34h[7:0]) is disabled. When visible, the value of 01h is the unique ID assigned to PCI power management capability by the PCI SIG.
Index 45h				**Next Item Pointer Register**
7:0	NextItemPointer	RO	00h	**Next Item Pointer:** This register specifies the pointer to the next capability item. This field returns 0 indicating that only one additional capability is supported.

Register Descriptions (Continued)

Table 4-6. PCI Configuration Register Definitions

Bit	Name	Access	Reset	Description
Index 46h-47h		**Power Management Capabilities Register**		
This register specifies capabilities related to PCI power management. This register is available only if the PCICapabilities bit (BAR1 Offset 00h[20]) is set in the nscControl register.				
15:11	pmeSupport	RO	01000	**Power Management Event Support:** Bit 14 is one indicating PME# may be asserted from the D3HOT power state. PME# is not capable of asserting from other states.
10	d2Support	RO	0	**D2 Power State Support:** This bit reads zero indicating that the D2 power state is not supported.
9	d1Support	RO	0	**D1 Power State Support:** This bit reads zero indicating that the D1 power state is not supported.
8	dynData	RO	0	**Dynamic Power Consumption Data:** This bit reads zero indicating that dynamic power consumption data is not provided.
7:6	RSVD	RO	0	**Reserved**
5	DSI	RO	0	**Driver Special Initialization:** This bit reads zero indicating that no special initialization is required beyond the standard PCI configuration header before a generic class driver is able to use the CS4210.
4	auxPower	RO	0	**Auxiliary Power:** This bit reads zero indicating that PME# generation in the D3COLD state is not supported.
3	pmeCLK	RO	0	**Power Management Event Clock:** This bit reads zero indicating that no host bus clock is required to generate PME#.
2:0	pmVerson	RO	000	**Power Management Version:** This field reads zero indicating compatibility with the PCI Bus Power Interface Management Specification previous to revision 1.0.
Note: The reset value is dependent upon the PCICapabilities bit in the nscControl register (BAR1+Offset 00h[20]) which can be configured by serial EEPROM. If enabled in the EEPROM the reset value of this register is 4000h. If not or if no EEPROM is present, then the reset value is 000h.				
Index 48h		**Power Management and Control Status Register**		
This register implements the control and status of the PCI power management function. This register is available only if the PCICapabilities bit (BAR1 Offset 00h[20]) is set in the nscControl register.				
15	pmeStatus	RC	1	**Power Management Event Status:** This bit is set when PME# is asserted. Write 1 to clear. The PME# signal is also cleared when this is bit is written to 1. A write of zero has no effect.
14:9	dynData	RO	000000	**Dynamic Power Consumption Data:** This field reads 0 indicating that dynamic data is not reported.
8	pmeEnab	R/W	0	**Power Management Event Enable:** When set, this bit enables the assertion of PME#.
7:5	RSVD	---	000	**Reserved**
4	dynDataEnab	RO		**Dynamic Power Consumption Data:** This bit reads 0 indicating that dynamic data is not reported.
3:2	RSVD	---	00	**Reserved**
1:0	pwrState	R/W		**Power State:** This field is used to determine and set the CS4210 power state. 00 = Current power state is D0 01 = Current power state is D1 10 = Current power state is D2 11 = Current power state is D3hot Since D1 and D2 are not supported, a write of either 01 or 10 is treated as the D0 state.
Note: The reset value is dependent upon the PCICapabilities bit in the nscControl register (BAR1+Offset 00h[20]) which can be configured by serial EEPROM. If enabled in the EEPROM the reset value of this register is 8000h. If not or if no EEPROM is present then the reset value is 000h.				

Register Descriptions (Continued)

Table 4-6. PCI Configuration Register Definitions

Bit	Name	Access	Reset	Description
Index 4Ah	**Power Management CSR Bridge Support Extension Register**			
7:0	pmcrBSE	RO	00h	**Power Management Control and Status Register Bridge Support Extension:** This field returns 0 indicating that the CS4210 does not support PCI-to-PCI bridging.
Index 4Bh	**Power Management Data Register**			
7:0	pmData	RO	00h	**Power Management Dynamic Power Data:** This field returns 0 indicating that the CS4210 does not report dynamic data.

Register Descriptions (Continued)

4.4 OHCI CONFIGURATION REGISTERS

The OHCI configuration registers are at the location specified by Base Address Register 0 (BAR0) in PCI configuration space.

The registers must be accessed as 32-bit entities with host processor quadlet reads or quadlet writes occurring on quadlet boundaries. When HCControl.LPS is 0, the only accessible registers are Version, VendorID, HCControl, GUID_ROM, GUIDHi and GUIDLo. Access to all other registers is undefined until HCControl.LPS is set to 1.

All register fields are initialized to zero or their default value upon power up. Reads of reserved fields yield undetermined results. Unless specified, a 1394 bus reset will not affect the register contents. The registers are either read/write or set/clear registers. The read/write registers are defined at a single location. The set/clear registers have one location for setting bits in the register and a second location for clearing those bits. When a value of 1 is written to a set location, that value is taken as a bit mask to update that bit. The other bits in the register are not changed. Writing a 1 to a clear location sets that bit to zero and will not change other bits. The register field descriptions in Table 4-7 describe the operating modes of those registers.

Table 4-8 is a map for the registers accessed through BAR0. Following this table are subsections providing detailed information for each register.

Table 4-7. Operating Modes

Access Tag	Name	Description
R	Read	Field may be read from the PCI bus.
W	Write	Field may be written from the PCI bus.
U	Update	Field may be autonomously updated by the OHCI hardware.
S	Set	Field may be set from the PCI bus.
C	Clear	Field may be cleared from the PCI bus.

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h-03h																**Version Register**															
RSVD							GUID_ROM	version								RSVD								revision							
04h-07h																**GUID_ROM Register**															
addrReset	RSVD					rdStart	RSVD	rdData								RSVD															
08h-0Bh																**ATRetries Register**															
secondLimit (not implemented)	cycleLimit (not implemented)															maxPhysResp-Retries			RSVD			maxATResp-Retries				maxATReq-Retries					
0Ch-0Fh																**CSRCompareData Register**															
csrData																															
10h-13h																**CSRCompareData Register**															
csrCompare																															
14h-17h																**CSRControlRegister**															
csrDone	RSVD																													csrSel	
18h-1Bh																**ConfigROMhdr Register**															
info_length																crc_length															
rom_crc_value																															

Register Descriptions (Continued)

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1Ch-1Fh	**BusID Register**																														
busID																															
20h-23h	**BusOptions Register**																														
irms	cmc	isc	bmc	pmc	RSVD			cyc_clk_acc								max_rec				RSVD				g		RSVD				link_spd	
24h-27h	**GUIDHi Register**																														
28h-2Bh	**GUIDLo Register**																														
node_vendor_ID																								chip_ID_Hi							
chip_ID_Lo																															
2Ch-30h	**Reserved**																														
34h-37h	**ConfigROMMap Register**																														
configROMaddr																						RSVD									
38h-3Bh	**PostedWriteAddressLo Register**																														
3Ch-3Fh	**PostedWriteAddressHi Register**																														
offsetLo																															
sourceID																				offsetHi											
40h-43h	**Vendor ID**																														
vendorUnique												vendorCompanyID																			
44h-4Fh	**Reserved**																														
50h-53h	**HCControl Set Register**																														
54h-57h	**HCControl Clear Register**																														
RSVD	noByteSwapData	RSVD							programPhyEnable	aPhyEnhanceEnable	RSVD			LPS	postedWriteEnable	linkEnable	softReset	RSVD												
58h-63h	**Reserved**																														
64h-67h	**Self-ID Buffer Pointer Register**																														
selfIDBufferPtr																															
RSVD																															
68h-6Bh	**Self-ID Count Register**																														
selfIDError	RSVD															selfIDGeneration															
RSVD																															
2selfIDSize																															
RSVD																															
6Ch-6Fh	**Reserved**																														
70h-73h	**IRMultiChanMaskHi Set Register**																														
74h-77h	**IRMultiChanMaskHi Clear Register**																														
IsochChannel[63:32]																															
78h-7Bh	**IRMultiChanMaskLo Set Register**																														
7Ch-7Fh	**IRMultiChanMaskLo Clear Register**																														

Register Descriptions (Continued)

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IsochChannel[31:0]																															
80h-83h 84h-87h **IntEvent Set Register IntEvent Clear Register**																															
RSVD					phyRegRcvd	cycleTooLong	unrecoverableError	cycleInconsistent	cycleLost	cycle64Seconds	cycleSynch	phy	RSVD	busReset	selfIDcomplete	RSVD							lockRespErr	postedWriteErr	isochRx	isochTx	RSPkt	RQPkt	ARRS	ARRQ	respTxComplete	reqTxComplete
88h-87h 8Ch-8Fh **IntMask Set Register IntMask Clear Register**																															
masterIntEnable	RSVD				phyRegRcvdIntEn	cycleTooLongIntEn	unrecoverableErrorIntEn	cycleInconsistentIntEn	cycleLostIntEn	cycle64SecondsIntEn	cycleSynchIntEn	phyIntEn	RSVD	busResetIntEn	selfIDcompleteIntEn	RSVD							lockRespErrIntEn	postedWriteErrIntEn	isochRxIntEn	isochTxIntEn	RSPktIntEn	RQPktIntEn	ARRSIntEn	ARRQIntEn	respTxCompleteIntEn	reqTxCompleteIntEn
90h-93h 94h-97h **IsochTxEvent Set Register IsochTxIntEvent Clear Register**																															
RSVD																								isochTxInt7	isochTxInt6	isochTxInt5	isochTxInt4	isochTxInt3	isochTxInt2	isochTxInt1	isochTxInt0
98h-9Bh 9Ch-9Fh **IsochTxIntMask Set Register IsochTxIntMask Clear Register**																															
RSVD																								isochTxIntMask7	isochTxIntMask6	isochTxIntMask5	isochTxIntMask4	isochTxIntMask3	isochTxIntMask2	isochTxIntMask1	isochTxIntMask0
A0h-A3h A4h-A7h **IsochRxIntEvent Set Register IsochRxIntEvent Clear Register**																															
RSVD																								isochRxInt7	isochRxInt6	isochRxInt5	isochRxInt4	isochRxInt3	isochRxInt2	isochRxInt1	isochRxInt0
A8h-ABh ACh-AFh **IsochRxIntMask Set Register IsochRxIntMask Clear Register**																															
RSVD																								isochRxIntMask7	isochRxIntMask6	isochRxIntMask5	isochRxIntMask4	isochRxIntMask3	isochRxIntMask2	isochRxIntMask1	isochRxIntMask0
B0h-DBh **Reserved**																															
DCh-DFh **Fairness Control Register**																															
RSVD																									pri_req						

Register Descriptions (Continued)

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
E0h-E3h E4h-E7h	**LinkControl Set Register LinkControl Clear Register**																														
RSVD									cycleSource	cycleMaster	cycleTimerEnable	RSVD									rcvPhyPkt	rcvSelfID	RSVD								
E8h-EBh	**Node ID and Status Register**																														
iDValid	root	RSVD		CPS	RSVD											busNumber															
ECh-EFh	**PHYControl Register**																														
rdDone	RSVD				rdAddr				rdData							rdReg	wrReg	RSVD			regAddr						wrData			
F0h-F3h	**IsochCycleTimer Register**																														
cycleSeconds												cycleCount																cycleOffset			
F4h-FFh	**Reserved**																														
100h-103h 104h-107h	**AsyncReqFilterHi Set Register AsyncReqFilterHi Clear Register**																														
asyncReqResourceAll	asyncReqResource[62:32]																														
108h-10Bh 10Ch-10Fh	**AsyncReqFilterLo Set Register AsyncReqFilterLo Clear Register**																														
asyncReqResource[31:0]																															
110h-113h 114h-117h	**PhysicalReqHi Set Register PhysicalReqHi Clear Register**																														
physReqResourceAll	physReqResource[62:32]																														
118h-11Bh 11Ch-11Fh	**PhysicalReqtLo Set Register PhysicalRequestLo Clear Register**																														
physReqResource[31:0]																															
120h-17Fh	**Reserved**																														

Register Descriptions (Continued)

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
180h-183h AsyncReqTxContextControl Set Register **184h-187h AsyncReqTxContextControl Clear Register**																															
RSVD																run	FSVD		wake	dead	active	RSVD					event code				
188h-18Bh Reserved																															
18Ch-18Fh AsyncReqTxCommandPtr Register																															
descriptorAddress																													Z		
190h-19Fh Reserved																															
1A0h-1A3h AsyncRespTxContextControl Set Register **1A4h-1A7h AsyncRespTxContextControl Clear Register**																															
RSVD																run	RSVD		wake	dead	active	RSVD					event code				
1A8h-1ABh Reserved																															
1ACh-1AFh AsyncRespTxCommandPtr Register																															
descriptoraddress																													Z		
1B0h-1BFh Reserved																															
1C0h-1C3h AsyncReqRxContextControl Set Register **1C4h-1C7h AsyncReqRxContextControl Clear Register**																															
RSVD																run	RSVD		wake	dead	active	RSVD		spd			event code				
1C8h-1CBh Reserved																															
1CCh-1CFh AsyncReqRxCommandPtr Register																															
descriptorAddress																													Z		
1D0h-1DFh Reserved																															
1E0h-1E3h AsyncRespRxContextControl Set Register **1E4h-1E7h AsyncRespRxContextControl Clear Register**																															
RSVD																run	RSVD		wake	dead	active	RSVD		spd			event code				
1E8h-1EBh Reserved																															
1ECh-1EFh AsyncRespRxCommandPtr Register																															
descriptorAddress																													Z		
1F0h-1FFh Reserved																															
200h-203h IsochTx0ContextControl Set Register **204h-207h IsochTx0ContextControl Clear Register**																															
cycleMatchEnable	cyclematch															run	RSVD		wake	dead	active	RSVD					event code				
208h-20Bh Reserved																															
20Ch-20Fh IsochTx0CommandPtr Register																															
descriptorAddress																													Z		

Register Descriptions (Continued)

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
210h-213h IsochTx1ContextControl Set Register **214h-217h IsochTx1ContextControl Clear Register**																															
cycleMatchEnable	cyclematch															run	RSVD		wake	dead	active	RSVD						event code			
218h-21Bh Reserved																															
21Ch-21Fh IsochTx1CommandPtr Register																															
descriptorAddress																														Z	
220h-223h IsochTx2ContextControl Set Register **224h-227h IsochTx2ContextControl Clear Register**																															
cycleMatchEnable	cyclematch															run	RSVD		wake	dead	active	RSVD						event code			
228h-22Bh Reserved																															
22Ch-22Fh IsochTx2CommandPtr Register																															
descriptorAddress																														Z	
230h-233h IsochTx3ContextControl Set Register **234h-237h IsochTx3ContextControl Clear Register**																															
cycleMatchEnable	cyclematch															run	RSVD		wake	dead	active	RSVD						event code			
238h-23Bh Reserved																															
23Ch-23Fh IsochTx3CommandPtr Register																															
descriptorAddress																														Z	
240h-243h IsochTx4ContextControl Set Register **244h-247h IsochTx4ContextControl Clear Register**																															
cycleMatchEnable	cyclematch															run	RSVD		wake	dead	active	RSVD						event code			
248h-24Bh Reserved																															
24Ch-24Fh IsochTx4CommandPtr Register																															
descriptorAddress																														Z	

Register Descriptions (Continued)

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
250h-253h IsochTx5ContextControl Set Register **254h-257h IsochTx5ContextControl Clear Register**																															
cycleMatchEnable	cyclematch															run	RSVD		wake	dead	active	RSVD						event code			
258h-25Bh Reserved																															
25Ch-25Fh IsochTx5CommandPtr Register																															
descriptorAddress																														Z	
260h-263h IsochTx6ContextControl Set Register **264h-267h IsochTx6ContextControl Clear Register**																															
cycleMatchEnable	cyclematch															run	RSVD		wake	dead	active	RSVD						event code			
268h-26Bh Reserved																															
26Ch-26Fh IsochTx6CommandPtr Register																															
descriptorAddress																														Z	
270h-273h IsochTx7ContextControl Set Register **274h-277h IsochTx7ContextControl Clear Register**																															
cycleMatchEnable	cyclematch															run	RSVD		wake	dead	active	RSVD						event code			
278h-27Bh Reserved																															
27Ch-27Fh IsochTx7CommandPtr Register																															
descriptorAddress																														Z	
280h-3FFh Reserved																															
400h-403h IsochRx0ContextControl Set Register **404h-407h IsochRx0ContextControl Clear Register**																															
bufferFill	isochheader	CycleMAtchEnable	multiChanmode	RSVD												run	RSVD		wake	dead	active	RSVD		spd				event code			
408h-40Bh Reserved																															
40Ch-40Fh IsochRx0CommandPtr Register																															
descriptorAddress																														Z	

Register Descriptions (Continued)

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
410h-413h																**IsochRx0ContextMatch Register**															
tag3	tag2	tag1	tag0	RSVD	cyclematch															sync				RSVD	tag1SyncFilter	channelNum					
414h-41Fh																**Reserved**															
420h-423h **424h-427h**																**IsochRx1ContextControl Set Register** **IsochRx1ContextControl Clear Register**														
bufferFill	isochheader	CycleMAtchEnable	multiChanmode	RSVD												run	RSVD		wake	dead	active	RSVD		spd		event code					
428h-42Bh																**Reserved**															
42Ch-42Fh																**IsochRx1CommandPtr Register**															
descriptorAddress																														Z	
430h-433h																**IsochRx1ContextMatch Register**															
tag3	tag2	tag1	tag0	RSVD	cyclematch															sync				RSVD	tag1SyncFilter	channelNum					
434h-43Fh																**Reserved**															
440h-443h **444h-447h**																**IsochRx2ContextControl Set Register** **IsochRx2ContextControl Clear Register**														
bufferFill	isochheader	CycleMAtchEnable	multiChanmode	RSVD												run	RSVD		wake	dead	active	RSVD		spd		event code					
448h-44Bh																**Reserved**															
44Ch-44Fh																**IsochRx2CommandPtr Register**															
descriptorAddress																														Z	
450h-453h																**IsochRx2ContextMatch Register**															
tag3	tag2	tag1	tag0	RSVD	cyclematch															sync				RSVD	tag1SyncFilter	channelNum					
454h-45Fh																**Reserved**															

Register Descriptions (Continued)

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
460h-463h 464h-467h	**IsochRx3ContextControl Set Register IsochRx3ContextControl Clear Register**																														
bufferFill	isochheader	CycleMAtchEnable	multiChanmode	RSVD												run	RSVD		wake	dead	active	RSVD		spd		event code					
468h-46Bh	**Reserved**																														
46Ch-46Fh	**IsochRx3CommandPtr Register**																														
descriptorAddress																													Z		
470h-473h	**IsochRx3ContextMatch Register**																														
tag3	tag2	tag1	tag0	RSVD	cyclematch															sync				RSVD	tag1SyncFilter	channelNum					
474h-47Fh	**Reserved**																														
480h-483h 484h-487h	**IsochRx4ContextControl Set Register IsochRx4ContextControl Clear Register**																														
bufferFill	isochheader	CycleMAtchEnable	multiChanmode	RSVD												run	RSVD		wake	dead	active	RSVD		spd		event code					
488h-48Bh	**Reserved**																														
48Ch-48Fh	**IsochRx4CommandPtr Register**																														
descriptorAddress																													Z		
490h-493h	**IsochRx4ContextMatch Register**																														
tag3	tag2	tag1	tag0	RSVD	cyclematch															sync				RSVD	tag1SyncFilter	channelNum					
494h-49Fh	**Reserved**																														
4A0h-4A3h 4A4h-4A7h	**IsochRx5ContextControl Set Register IsochRx5ContextControl Clear Register**																														
bufferFill	isochheader	CycleMAtchEnable	multiChanmode	RSVD												run	RSVD		wake	dead	active	RSVD		spd		event code					
4A8h-4ABh	**Reserved**																														
4ACh-4AFh	**IsochRx5CommandPtr Register**																														
descriptorAddress																													Z		

Register Descriptions (Continued)

Table 4-8. OHCI Configuration Register Map/Summary: BAR0+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
4B0h-4B3h	**IsochRx5ContextMatch Register**																														
tag3	tag2	tag1	tag0	RSVD	cyclematch															sync				RSVD	tag1SyncFilter	channelNum					
4B4h-4BFh	**Reserved**																														
4C0h-4C3h **4C4h-4C7h**	**IsochRx6ContextControl Set Register** **IsochRx6ContextControl Clear Register**																														
bufferFill	isochheader	CycleMAtchEnable	multiChanmode	RSVD												run	RSVD		wake	dead	active	RSVD		spd		event code					
4C8h-4CBh	**Reserved**																														
4CCh-4CFh	**IsochRx6CommandPtr Register**																														
descriptorAddress																														Z	
4D0h-4D3h	**IsochRx6ContextMatch Register**																														
tag3	tag2	tag1	tag0	RSVD	cyclematch															sync				RSVD	tag1SyncFilter	channelNum					
4D4h-4DFh	**Reserved**																														
4E0h-4E3h **4E4h-4E7h**	**IsochRx7ContextControl Set Register** **IsochRx7ContextControl Clear Register**																														
bufferFill	isochheader	CycleMAtchEnable	multiChanmode	RSVD												run	RSVD		wake	dead	active	RSVD		spd		event code					
4E8h-4EBh	**Reserved**																														
4ECh-4EFh	**IsochRx6CommandPtr Register**																														
descriptorAddress																														Z	
4F0h-4F3h	**IsochRx7ContextMatch Register**																														
tag3	tag2	tag1	tag0	RSVD	cyclematch															sync				RSVD	tag1SyncFilter	channelNum					
4F4h-4FFh	**Reserved**																														

Register Descriptions (Continued)

4.4.1 Version Register

The Version register (Table 4-9) contains a 32-bit value which indicates the version and capabilities of the interface. The register is expected to be used to indicate the level of functionality present in the CS4210.

4.4.2 GUIDROM Register

The Global Unique ID ROM register (Table 4-10) is used to access the GUID ROM. To initialize the GUID ROM read address, software sets GUIDROM.addrReset to one. Once software detects that GUIDROM.addrReset is zero, indicating that the reset has completed, then software may set GUIDROM.rdStart to read a byte. Upon the completion of each read, the CS4210 places the read byte into GUIDROM.rdData, advances the GUID ROM address by one byte to set up for the next read, and clears GUIDROM.rdStart to 0 to indicate to software that the requested byte has been read.

Table 4-9. BAR0+Offset 00h: Version Register

Bit	Name	Access	Reset	Description
31:25	RSVD	--	0	**Reserved**
24	GUID_ROM	R	N/A	**Global Unique ID ROM:** The third and fourth quadlets of the Bus_Info_Block are automatically loaded on hardware reset.
23:16	version	R	01h	**Major Version:** This field contains the BCD encoded value representing the major version of the highest numbered 1394 OpenHCI specification with which the CS4210 is compliant.
15:8	RSVD	--	0	**Reserved**
7:0	revision	R	0	**Minor Version:** This field contains the BCD encoded value representing the minor version of the highest numbered 1394 OpenHCI specification with which the CS4210 is compliant.

Table 4-10. BAR0+Offset 04h: GUID ROM Register

Bit	Name	Access	Reset	Description
31	addrReset	RSU	0	**Address Reset:** Software sets this bit to one to reset the GUID ROM address to zero. When the CS4210 completes the reset, it clears addrReset to zero. Upon resetting the GUID ROM address, the CS4210 does *not* automatically fill rdData with the data from byte address 0.
30:26	RSVD	--	0	**Reserved**
25	rdStart	RSU	0	**Read Start**: A read of the currently addressed GUID ROM byte is started on the transition of this bit from a zero to a one. When the CS4210 completes the read, it clears rdStart to zero and advances the GUID ROM byte address by one byte.
24	RSVD	--	0	**Reserved**
23:16	rdData	RU	Undef	**Read Data:** The data read from the GUID ROM.
15:0	RSVD	--	0	**Reserved**

Register Descriptions (Continued)

4.4.3 ATRetries Register

The ATRetries register (Table 4-11) holds the number of times the CS4210 attempts to do a retry for asynchronous DMA request transmit and for asynchronous physical and DMA response transmit. A packet may only be retried when a “busy” acknowledge or ack_data_error is received from the target node, including ack_data_error’s resulting from FIFO underflows. A packet is not retried under any other circumstance, including receipt of evt_missing_ack.

Table 4-11. BAR0+Offset 08h: ATRetries Register

Bit	Name	Access	Reset	Description
31:29	secondLimit	R	0	**Second Limit:** Not implemented.
28:16	cycleLimit	R	0	**Cycle Limit:** Not implemented.
15:11	maxPhysRespRe-tries	RW	Undef	**Maximum Physical Response Retries:** This field tells the response unit within the RDMA module how many times to attempt to retry the transmit operation for the response packet. Note that this value is used only for responses to physical requests. If the retry count expires for a physical response, the packet is discarded by the CS4210. Software is not notified.
10:8	RSVD	--	0	**Reserved**
7:4	maxATRespRetries	RW	Undef	**Maximum AT Physical Response Retries:** This field tells the asynchro-nous transmit response unit within the ATDMA module how many times to attempt to retry the transmit operation for a software transmitted (non-phys-ical) asynchronous response packet.
3:0	maxATReqRetries	RW	Undef	**Maximum AT Request Retries:** This field tells the asynchronous transmit request unit within the ATDMA how many times to attempt to retry the trans-mit operation for an asynchronous request packet.

Register Descriptions (Continued)

4.4.4 Autonomous CSR Resources

The CS4210 implements a number of autonomous CSR resources. In particular the 1394 compare-swap bus management registers are implemented in hardware, as is the config ROM header, the Bus_Info_Block and access to the first 1 KB of the configuration ROM. The DMA units handle external 1394 bus requests to these resources automatically. The serial bus registers shown in Table 4-12 manage this function for the local host. 1394 requires certain 1394 bus management resource registers be accessible only via “quadlet read” and “quadlet lock” (compare-and-swap) transactions, otherwise ack_type_error is sent. These special bus management resource registers are implemented internal to the CS4210 to allow atomic compare-and-swap access from either the host system or from the 1394 bus.

When these serial bus management resource registers are accessed from the 1394 bus, the atomic compare-and-swap transaction is autonomous, without software intervention. If ack_complete is not received to end the transaction for the generated lock response, IntEvent.lockRespErr (refer to Table 4-26 on page 70) is triggered. To access these bus management resource registers from the host, the registers shown in Table 4-13 are used. To access these bus management resource registers from the host bus, first load the CSRReadData register with the new data value to be loaded into the appropriate resource. Then load the CSRCompareData register with the expected value. Finally, write the CSRControl register with the selector value of the resource. A write to the CSRControl register initiates a compare-and-swap operation on the selected resource. When the compare-and-swap operation is complete, the CSRControl register csrDone bit is set, and the CSRReadData register contains the value of the selected resource prior to the host initiated compare-and-swap operation. Note that an arbitrary update of these resources cannot be done. Only compare-and-swap operations can be used to modify the contents of these internal resource registers.

Table 4-12. Serial Bus Registers

CSR Address	csrSel Bits (BAR0+Offset 14h[1:0]	Description	1394-1995 Section #	Reset (Hardware or Bus Reset)
FFFF_F000_021Ch	00	BUS_MANAGER_ID	8.3.2.3.6	03Fh
FFFF_F000_0220h	01	BANDWIDTH_AVAILABLE	8.3.2.3.7	1333h
FFFF_F000_0224h	10	CHANNELS_AVAILABLE_HI	8.3.2.3.8	FFFF_FFFFh
FFFF_F000_0228h	11	CHANNELS_AVAILABLE_LO	8.3.2.3.8	FFFF_FFFFh

Table 4-13. CSR Registers

Bit	Name	Access	Reset	Description
BAR0+Offset 0Ch				**CSRReadData Register**
31:0	csrData	RWU	Undef	**Control/Status Register Read Data:** At start of operation, the data to be stored if the compare is successful.
BAR0+Offset 10h				**CSRCompareData Register**
31:0	csrCompare	RW	Undef	**Control/Status Register Compare:** The data to be compared with the existing value of the CSR resource.
BAR0+Offset 14h				**CSRControl Register**
31	csrDone	RU	1	**Control/Status Register Done:** This bit is set when a compare-swap operation is completed. It is reset whenever this register is written.
30:2	RSVD	--	0	**Reserved**
1:0	csrSel	RW	Undef	**Control/Status Register Resource Selection:** 00 = BUS_MANAGER_ID 01 = BANDWIDTH_AVAILABLE 10 = CHANNELS_AVAILABLE_HI 11 = CHANNELS_AVAILABLE_LO

Register Descriptions (Continued)

4.4.5 Configuration ROM Header Register

The configuration ROM header register (Table 4-14) is a 32-bit number that externally maps to the 1st quadlet of the 1394 configuration ROM (1394 address at offset FFFF_F000_0400h). This register is written locally at BAR0+Offset 18h.

4.4.6 Bus Identification Register

The Bus Identification register (Table 4-15) is a 32-bit number that externally maps to the first quadlet of the Bus_Info_Block.

Table 4-14. BAR0+Offset 18h: ConfigROMhdr Register

Bit	Name	Access	Reset	Description
31:24	info_length	RWU	00h	**Information Length:** IEEE 1394 bus management field. Must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
23:!6	crc_length	RWU	00h	**Cyclical Redundancy Check Length:** IEEE 1394 bus management field. Must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
15:0	rom_crc_value	RWU	0000h	**ROM Cyclical Redundancy Check Value:** IEEE 1394 bus management field. Must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.

Table 4-15. BAR0+Offset 1Ch: Bus Identification Register

Bit	Name	Access	Reset	Description
31:0	busID	R	31333934h	**Bus Identification:** This 32-bit number externally maps to the first quadlet of the Bus_Info_Block. It contains the constant 31333934h which is the ASCII value for “1394”.

Register Descriptions (Continued)

4.4.7 Bus Options Register

The Bus Options register (Table 4-16) is a 32-bit number that externally maps to the second quadlet of the Bus_Info_Block. This register is written locally at BAR0+Offset 20h. This register is loaded from the serial EEPROM, if present, which sets the values after a hardware reset.

Table 4-16. BAR0+Offset 20h: Bus Options Register

Bit	Name	Access	Reset	Description
31:27	irmc	RW	Undef	**Isochronous Resource Manager Capable:** IEEE 1394 bus management field. Must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
30	cmc	RW	Undef	**Cycle Manager Capable:** IEEE 1394 bus management bit. Must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
29	isc	RW	Undef	**Isochronous Capable:** IEEE 1394 bus management bit. Must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
28	bmc	RW	Undef	**Bus Manager Capable:** IEEE 1394 bus management bit. Must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
27	pmc	RW	Undef	**Power Manager Capable:** IEEE 1394 bus management bit. Must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
26:24	RSVD	--	0	**Reserved**
23:16	cyc_clk_acc	RW	Undef	**Cycle Clock Access:** IEEE 1394 bus management field. Must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
15:12	max_rec	RW	**	**Maximum Received:** IEEE 1394 bus management field. Hardware initializes max_rec to 1024 if no serial EEPROM is used or to the value stored for this register in the EEPROM if present. Software may change max_rec, however, this field must be valid at any time the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set to 1. Note that received block write request packets with a length greater than max_rec shall generate an ack_type_error if the request is not handled by the physical response unit, and may generate an ack_type_error otherwise. ** Reset values: For a hardware reset, max_rec is set 1024 or to the value stored in the serial EEPROM, if present. For a soft reset, max_rec is not changed.
11:8	RSVD	--	0	**Reserved**
7:6	g	RW	Undef	**Generation Counter:** This field increments if any portion of configuration ROM has changed since the prior bus reset.
5:3	RSVD	--	0	**Reserved**
2:0	link_spd	RWU RU	**	**Link Speed:** 000 = 100 Mbits/sec; 001 = 200 Mbits/sec; 010 = 400 Mbits/sec; all other values are reserved. **On a hardware reset, link_spd is set 010 (400 Mbits/sec) or to the value stored in the serial EEPROM, if present. If the link_spd write protect in the nscControl register (BAR1+Offset 00h[12]) is not set, software is permitted to change link_spd to a lower value, which causes the link to reject packets arriving at higher speeds. **On a software reset, the value of link_spd is undefined.

Register Descriptions (Continued)

4.4.8 Global Unique ID Register

The Global Unique ID (GUID) (Table 4-17) is a 64-bit number that externally maps to the third and fourth quadlets of the Bus_Info_Block. The GUID is contained in the two 32-bit registers, GUIDHi and GUIDLo, BAR0+Offset 24h-28h. The vendor ID is obtained from the IEEE Registration Authority Committee (RAC). A company does not need to obtain a vendor ID if it has been previously assign an IEEE 48-bit Globally Assigned Address Block or an IEEE-assigned Organizationally Unique Identifier for use in network applications.

4.4.9 Configuration ROM Mapping Register

The Configuration ROM Mapping register (Table 4-18) contains the start address within system bus space that maps to the start address of the 1394 configuration ROM for this node. Only quadlet reads to the first 1 KB of the configuration ROM map to system bus space, all other transactions to this space are rejected with a 1394 “ack_type_error”. Since the low order 10 bits of this address are reserved and assumed to be zero, the system address for the configuration ROM must start on a 1 KB boundary. Note that the first five quadlets of the 1394 configuration ROM space are mapped to the configuration ROM header and the Bus_Info_Block, and so are handled directly by the CS4210. This means that the first five quadlets addressed by the configuration ROM mapping register are not used. Software should ensure this address is valid before setting HCControl.linkEnable (BAR0+Offset 50h[17]) to one.

Table 4-17. GUID Register

Bit	Name	Access	Reset	Description
BAR0+Offset 24h				**GUIDHi Register**
31:8	node_vendor_ID	RW	**	**Vendor ID Code**: IEEE 1394 bus management field. Must be set by firmware or hardware before the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
7:0	chip_ID_Hi	RW		**Chip Identification High:** The upper 8 bits of the chip ID. IEEE 1394 bus management field. Must be set by firmware or hardware before the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.
BAR0+Offset 28h				**GUIDLo Register**
31:0	chip_ID_Lo	RW	**	**Chip Identification Low**: The lower 32 bits of the chip ID. IEEE 1394 bus management field. Must be set by firmware or hardware before the HCControl.linkEnable bit (BAR0+Offset 50h[17]) is set.

**The Global Unique ID (GUID) register is reset to 0 after a host power (hardware) reset. A value of 0 is an illegal value. This register is not affected by a software reset. The GUID register is written only once after host power reset, by either:

1) an autonomous load operation from a local, un-modifiable such as the serial EEPROM or local parallel ROM, or

2) a single host write to each field performed only by firmware that is always executed on a hardware reset which affects the CS4210. This firmware, as well as the GUID value that is loaded, may not be modifiable by any user action.

After one of these load mechanisms has executed, the GUID register is read-only.

Table 4-18. BAR0+Offset 34h: ConfigROMMap Register

Bit	Name	Access	Reset	Description
31:10	configROMaddr	RW	Undef	**Configuration ROM Address:** If a quadlet read request to 1394 offset FFFF_F000_0400h through offset FFFF_F000_07FFh is received, then the low order 10 bits of the offset are added to this register to determine the host memory address of the returned quadlet.
9:0	RSVD	--	0	**Reserved**

Register Descriptions (Continued)

4.4.10 PostedWriteAddress Register

The PostedWriteAddress register (Table 4-19) is a 64-bit number which indicates the bus and node numbers (source ID) of the node that issued the write that failed, and the address that node attempted to access. The PostedWriteAddress is contained in two 32-bit registers, PostedWriteAddressHi and PostedWriteAddressLo. The IntEvent.PostedWriteErr bit (BAR0+Offset 80h[8]) allows hardware to generate an interrupt when a write fails.

The PostedWriteAddress register points to a queue in the CS4210. This queue is accessed by software through the PostedWriteAddress register. When a posted write fails, its address and node's source ID are placed in this queue, and the interrupt is generated. In addition, that packet is removed from the FIFO. By removing the packet from the FIFO, the CS4210 is not blocked from performing future transactions on the 1394 and host buses. When software reads from these registers, that entry is removed from the queue, the next address and source ID are placed at the head of the queue, and another interrupt is generated. When the queue is empty, the CS4210 stops generating interrupts. In order to guarantee the accuracy of the Posted Write error registers, software must perform the following algorithm when the posted write error interrupt is encountered:

1) Read the PostedWriteAddressHi.offsetHi field.
2) Read the PostedWriteAddressLo.offsetLo field.
3) Clear the IntEvent.PostedWriteError bit (BAR0+Offset 80h[8]).

This guarantees that software receives all information it requires about the first posted write, allowing another interrupt to be generated for future posted writes, and simplifies the CS4210 hardware. The CS4210 does not monitor that all three events occur before it moves to the next item in the queue. It considers the information read once it sees the IntEvent.PostedWriteError bit (BAR0+Offset 80h[8]) cleared to 0.

4.4.11 Vendor ID Register

The Vendor ID register holds the company ID of National Semiconductor Corporation indicating that additional registers have been specified in the CS4210.

Table 4-19. PostedWriteAddress Register

Bit	Name	Access	Reset	Description
BAR0+Offset 38h			**PostedWriteAddressLo Register**	
31:0	offsetLo	RU	Undef	**Offset Low:** The low 32-bits of the 1394 destination offset of the write request that failed.
BAR0+Offset 3Ch			**PostedWriteAddressHi Register**	
31:16	sourceID	RU	Undef	**Source ID:** The busNumber and nodeNumber of the node that issued the write request that failed.
15:0	offsetHi	RU	Undef	**Offset High:** The upper 16-bits of the 1394 destination offset of the write request that failed.

Table 4-20. BAR0+Offset 40h: Vendor ID Register

Bit	Name	Access	Reset	Description
31:24	vendorUnique	R	0	**Vendor Unique:** 0h
23:0	vendorCompanyID	R	80017h	**Vendor Company Identification:** The company ID National Semiconductor Corporation of 80017h.

Register Descriptions (Continued)

4.4.12 HCControl Register

The HCControl register (Table 4-21) provides flags for controlling the CS4210. There are two addresses for this register:

1) BAR0+Offset 50h: HCControl Set

2) BAR0+Offset 54h: HCControl Clear

On read, both addresses return the contents of the control register. For writes, the two addresses have different behavior: a one bit written to HCControl Set causes the corresponding bit in the HCControl register to be set, while a zero bit leaves the corresponding bit in the HCControl register unaffected. On the other hand, a one bit written to HCControl Clear causes the corresponding bit in the HCControl Set register to be cleared, while a zero bit leaves the corresponding bit in the HCControl Set register unaffected.

Table 4-21. BAR0+Offset 50h (Set) and 54h (Clear): HCControl Register

Bit	Name	Access	Reset	Description
31	RSVD	--	0	**Reserved**
30	noByteSwapData	RSC	Undef	**No Byte Swap Data:** This bit is used to control whether physical accesses to locations outside the CS4210 itself as well as any other DMA data accesses should be swapped or not. When 0, data quadlets are sent/ received in little endian order. When 1, data quadlets are sent/received in big endian order. See Section 4.4.12.1 "noByteSwapData" on page 66 for further information. Software should change this bit only when linkEnable (bit 17) is 0, otherwise unspecified behavior results.
29:24	RSVD	--	0	**Reserved**
23	programPhyEnable	RC	1	**Program PHY Enable:** This bit informs upper-level generic software (e.g., OHCI device driver) if lower-level implementation specific software (e.g., BIOS or Open Firmware) has consistently configured P1394a enhancements in the CS4210 and CS4103. If the implementation does not support P1394a enhancements, lower-level implementation specific software must clear this bit. When 1 and while linkEnable (bit 17) is 0, generic software is responsible for configuring the P1394a enhancements within the CS4103 and the aPhyEnhanceEnable bit within the CS4210 Link in a consistent manner. When 0, generic software may not modify the P1394a enhancement configuration in either the CS4210 or CS4103 and cannot interpret the setting of aPhyEnhanceEnable. A soft reset and a bus reset do not affect this bit.
22	aPhyEnhanceEnable	RSC	0	**A PHY Enhancement Enable:** When the programPhyEnable bit is 1, this bit is used by generic, implementation independent software (e.g., OHCI device driver) to enable the CS4210 Link to use all of P1394a enhancements. Generic software can only modify this bit when the programPhyEnable bit is 1 and the linkEnable (bit 17) bit is 0. This bit is meaningless to software when the programPhyEnable bit is 0. When 0, none of the P1394a enhancements are enabled within the Link. When 1, the set of all P1394a enhancements is enabled within the Link. A soft reset and a bus reset do not affect this bit. See Section 4.4.12.2 "programPhyEnable and aPhyEnhanceEnable" on page 67 for more information.
21:20	RSVD	--	0	**Reserved**
19	LPS	RS	0	**Link Power Status:** Software must set LPS to 1 to permit CS4210/CS4103 communication. Once set, the link can use LREQs to perform CS4103 reads and writes. An LPS value of 0 prevents CS4210/CS4103 communication. In this state, the only accessible CS4210 registers are Version, VendorID, HCControl, GUID_ROM, GUIDHi and GUIDLo. Access to other registers is not defined. Hardware and software resets clear LPS to 0. Software shall not clear LPS. See the Section 4.4.12.3 "LPS and linkEnable" on page 67 for more information.

Register Descriptions (Continued)

Table 4-21. BAR0+Offset 50h (Set) and 54h (Clear): HCControl Register (Continued)

Bit	Name	Access	Reset	Description
18	postedWriteEnable	RSC	Undef	**Posted Write Enable:** This bit is used to enable (1) or disable (0) physical posted writes. When disabled (0), physical writes are handled but are posted and instead are ack'ed with ack_pending. Software should change this bit only when linkEnable (bit 17) is 0, otherwise unspecified behavior results. See Section 3.6 "Physical Requests" on page 26 for information about posted writes.
17	linkEnable	RSU	0	**Link Enable:** Software must set this bit to 1 when the system is ready to begin operation and then force a bus reset. This bit is necessary to keep other nodes from sending transactions before the local system is ready. When this bit is clear the CS4210 is logically and immediately disconnected from the 1394 bus. The link will not process or interpret any packets received from the CS4103, nor will it generate any bus requests. However, the link will access CS4103 registers via the CS4103 control register. This bit is cleared to 0 by a hardware reset or software reset, and must not be cleared by software. Software must not set the linkEnable bit until the Configuration ROM mapping register (Section 4.4.9 on page 63) is valid.
16	softReset	RSU	**	**Soft Reset:** When set to 1, the CS4210's state is reset, all FIFO's are flushed and all CS4210 OHCI registers are set to their hardware reset values unless otherwise specified. Registers outside of the OpenHCI realm (i.e., PCI and NSC defined registers) are not affected. **The read value of this bit is 1 while a soft reset or a hard reset is in progress. The read value of this bit is 0 when neither a soft reset nor hard reset are in progress. Software can use the value of this bit to determine when a reset has completed and the CS4210 is safe to operate.
15:0	RSVD	--	0	**Reserved**

4.4.12.1 noByteSwapData

The 1394 bus is quadlet based big endian. By convention, when quadlets are sent in big endian order, the leftmost byte (bits [31:24]) of a quadlet is sent first. When sent in little endian order, the right most byte (bits [7:0]) is sent first with the leftmost bit of each byte sent first.

When the CS4210 sends/receives a packet, the header information is always sent/received in big endian order (leftmost byte first). Header information is composed of a sequence of quadlets which is invariant over big and little endian system.

When the HCControl.noByteSwapData bit is not set, data quadlets are sent/received in little endian order and when HCControl.noByteSwapData is set, data quadlets are sent/received in big endian order. The data quadlets as classified by the OHCI transaction codes (tcodes) that are subject to swap are:

1) Any data quadlet covered by data CRC (tcodes 1h, 7h, 9h, Ah, and Bh).
2) The data quadlet in a quadlet write request (tcode 0h).
3) The data quadlet in a quadlet read response (tcode 6h).

Since the cycle_time is self contained within the CS4210, it is never byte-swapped regardless of the setting of the noByteSwapData bit.

The data in a PHY packet (identified internally with tcode Eh) is not byte swapped for send or receive.

Note: Due to some confusion regarding this bit, an explanation and some examples are available on the OpenHCI FTP site.

Register Descriptions (Continued)

4.4.12.2 programPhyEnable and aPhyEnhanceEnable

After a hardware or software reset, system software must ensure that the CS4210 and the CS4103 are set to a consistent, compatible set of P1394a enhancements. The programPhyEnable and aPhyEnhanceEnable bits are provided to enable software to accomplish this task. Since different levels of software may be responsible for ensuring this setup, the programPhyEnable bit is defined to allow communication between implementation specific lower-level software (e.g., BIOS or Open Firmware) and generic, implementation independent upper-level software (e.g., OHCI device driver). If generic software reads this bit as a 1, it is responsible for configuring the P1394a enhancements in both the CS4210 and CS4103 in a consistent manner (either all enhancements enabled or all enhancements disabled). A 0 value for this bit informs the upper-level system software that no further changes to the P1394a configurations of the CS4210 and CS4103 are permitted since either:

1) Lower-level software has previously performed initialization appropriate to the CS4210 capabilities, or
2) The link has hardwired P1394a capabilities to match the CS4103. Note that this bit is only a software flag and does not control any CS4210 functionality.

The programPhyEnable bit may be read-only, returning a zero value, if upper-level software is not involved in the configuration of P1394a enhancements for the CS4210 and CS4103. This is appropriate when the CS4210 and CS4103 are hardwired with compatible settings or when lower-level software consistently configures both the CS4210 and CS4103. To allow the possibility for upper-level software control of P1394a enhancements, programPhyEnable should be implemented as read/clear with a hardware reset value of 1. Software should clear programPhyEnable once the CS4210 and CS4103 have been programmed consistently by either lower-level or upper-level software. When programPhyEnable is set to 1, the aPhyEnhanceEnable bit allows generic software to enable or disable all P1394a enhancements within the CS4210 Link. A value of 1 for aPhyEnhanceEnable configures the Link to use all P1394a enhancements and is appropriate when software has enabled all of the enhancements within the CS4103. Likewise, a value of 0 prevents the Link from using any P1394a enhancements and is appropriate when software has disabled all of the enhancements within the CS4103. Note that generic software must not attempt to modify or interpret the setting of the aPhyEnhanceEnable bit if programPhyEnable contains a 0. The aPhyEnhanceEnable bit is read/set/clear and it resets to 0 for default compatibility with legacy PHYs. These bits are accessible from the nscControl register (BAR1+Offset 00h[14,13]). The aPhyEnhanceEnable bit can be initialized with the serial EEPROM.

4.4.12.3 LPS and linkEnable

There are three basic tasks and ensuing requirements with respect to the Phy-Link interface:

1) Bootstrap of Open HCI.
   This requires a mechanism to configure the CS4210 and CS4103 prior to receiving any packets or generating any bus requests.
2) Recovery from a hung system.
   This requires a mechanism which places OpenHCI in a near pre-bootstrap condition, and allows the CS4210 and CS4103 to get back into sync if required.
3) Power Management via Suspend/Resume
   This requires a mechanism to inform the CS4103 that Phy-Link communication is no longer required and the CS4103 can suspend itself if no active ports remain.

To achieve proper behavior in satisfying these requirements, software shall always assert the signals in the following sequence: LPS, then linkEnable, then any other individual context enables or runs. The CS4210 behavior when violating this order is undefined and can produce unreliable behavior. Table 4-22 illustrates the progressive functionality as these signals are asserted.

Following a hardware or software reset, LPS and linkEnable are Off as shown in Step a (in Table 4-22). Software proceeds to enable the link power status (b) and when SCLK has started, software can configure the CS4210 and CS4103 registers as listed in Step c (e.g., Self-ID receive DMA registers). Setting linkEnable in step d enables some DMA function, and asserting contextControl.run (e) for the CS4210 contexts then yields full functionality.

Table 4-22. LPS and linkEnable Assertion

Step	LPS (BAR0+Offset 50h[19])	linkEnable (BAR0+Offset 50h[17])	contextControl.run	Sequence Comments
a	Off	Off	Off	Initial State
b	On	Off	Off	Allows SCLK to start
c	On	Off	Off	Config Phy-Link registers
d	On	On	Off	Initiate Bus Reset
e	On	On	Off	Physical DMA/Cycle Starts Okay
f	On	On	On	Normal Operation

Register Descriptions (Continued)

4.4.13 Self-ID Buffer Pointer Register

The Self-ID Buffer Pointer register (Table 4-23) points to the buffer the Self-ID packets are DMA'ed into during bus initialization.

4.4.14 Self-ID Count Register

The Self-ID Count register (Table 4-24) keeps a count of the number of times the bus Self-ID process has occurred, flags Self-ID packet errors and keeps a count of the amount of Self-ID data in the Self-ID buffer.

Table 4-23. BAR0+Offset 64h: Self-ID Buffer Pointer Register

Bit	Name	Access	Reset	Description
31:11	selfIDBufferPtr	RW	Undef	**Self-ID Buffer Pointer:** Contains the 2 KB aligned base address of the buffer in host memory where received Self-ID packets are stored. The contents of this field are undefined after a chip reset.
10:0	RSVD	--	0	**Reserved**

Table 4-24. BAR0+Offset 68h: Self-ID Count Register

Bit	Name	Access	Reset	Description
31	selfIDError	RU	Undef	**Self-ID Error:** When this bit is one, an error was detected during the most recent Self-ID packet reception. The contents of the Self-ID buffer are undefined. This bit is cleared after a Self-ID reception in which no errors are detected. Note that an error can be a hardware error or a host bus write error.
30:24	RSVD	--	0	**Reserved**
23:16	selfIDGeneration	RU	Undef	**Self-ID Generation:** The value in this field increments each time a bus reset is detected. This field rolls over to 0 after reaching 255.
15:13	RSVD	--	0	**Reserved**
12:2	2selfIDSize	RU	Undef	**Quadlet Self-ID Size:** This field indicates the number of quadlets that have been written into the Self-ID buffer for the current Self-ID Generation. This includes the header quadlet and the Self-ID data. This field is cleared to zero as soon as a bus reset is detected.
1:0	RSVD	--	0	**Reserved**

Register Descriptions (Continued)

4.4.15 IRMultiChanMask Registers

An isochronous channel mask is used to enable packet receives from up to 64 specified isochronous data channels. Software enables receives for any number of isochronous channels by writing ones to the corresponding bits in the:

1) BAR0+Offset 70h: IRMultiChanMaskHi Set

2) BAR0+Offset 78h: IRMultiChanMaskLo Set

To disable receives for any isochronous channels, software writes ones to the corresponding bits in the:

1) BAR0+Offset 74h: IRMultiChanMaskHi Clear

2) BAR0+Offset 7Ch: IRMultiChanMaskLo Clear

A read of each IRMultiChanMask register shows which channels are enabled; a one for enabled, a zero for disabled. The IRMultiChanMask registers are not changed by a bus reset. The state of these registers is undefined following a hard reset or soft reset.

Table 4-25. IRMultChanMask Registers

Bit	Name	Access	Reset	Description
BAR0+Offset 70h			**IRMultiChanMaskHi Set Register**	
31:0	IsochChannel[63:32]	RSC	Undef	**Isochronous Channels [63:32]:** Bits [31:0] correspond to channels [63:32]. Set to one to enable receives to the corresponding channel.
BAR0+Offset 74h			**IRMultiChanMaskHi Clear Register**	
31:0	IsochChannel[63:32]	RSC	Undef	**Isochronous Channels [63:32]:** Bits [31:0] correspond to channels [63:32]. Set to one to disable receives to the corresponding channel.
BAR0+Offset 78h			**IRMultiChanMaskLo Set Register**	
31:0	IsochChannel[31:0]	RSC	Undef	**Isochronous Channels [31:0]:** Bits [31:0] correspond to channels [31:0]. Set to one to enable receives to the corresponding channel.
BAR0+Offset 7Ch			**IRMultiChanMaskLo Clear Register**	
31:0	IsochChannel[31:0]	RSC	Undef	**Isochronous Channels [31:0]:** Bits [31:0] correspond to channels [31:0]. Set to one to disable receives to the corresponding channel.

Register Descriptions (Continued)

4.4.16 Interrupts

The CS4210 reports two classes of interrupts to the host: DMA interrupts and device interrupts. DMA interrupts are generated when DMA transfers complete (or are aborted). Device interrupts come directly from the remaining 1394 Open HCI logic. For example, one of these interrupts could be sent in response to the asserting edge of cycleStart, a signal which indicates that a new isochronous cycle has started.

The CS4210 contains two primary 32-bit registers to report and control interrupts: IntEvent and IntMask. Both registers have two addresses: a “Set” address and a “Clear” address. For a write to either register, a “one” bit written to the “Set” address causes the corresponding bit in the register to be set (excluding bits which are read-only), while a “one” bit written to the “Clear” address causes the corresponding bit to be cleared. For both addresses, writing a “zero” bit has no effect on the corresponding bit in the register.

The IntEvent register contains the actual interrupt request bits. Each of these bits corresponds to either a DMA completion event, or a transition on a device interrupt line. The IntMask register is ANDed with the IntEvent register to enable selected bits to generate processor interrupts. Software writes to the IntEvent Clear register to clear interrupt conditions reported in the IntEvent register.

A processor interrupt is generated when one or more unmasked bits are set in the IntEvent register. Low-level software responds to the interrupt by reading the IntEvent register, then writing the value read to the IntEvent Clear register. At this point the interrupt request is deasserted (assuming no new interrupt bit has been set). Software can proceed to process the reported interrupts in whatever priority order it chooses, and is free to re-enable interrupts as soon as the IntEvent Clear register is written.

In addition, the CS4210 contains four secondary 32-bit registers to report and control interrupts for isochronous transmit and receive contexts. Each register has two addresses: a “Set” address and a “Clear” address.

4.4.16.1 IntEvent Register

This register reflects the state of the various interrupt sources from the 1394 Open HCI. The interrupt bits are set by an asserting edge of the corresponding interrupt signal, or by software by writing a one to the corresponding bit in the IntEvent Set register. They are cleared by writing a one to the corresponding bit in the IntEvent Clear register.

Reading the IntEvent Set register (BAR0+Offset 80h) returns the current state of the IntEvent register. Reading the IntEvent Clear register (BAR0+Offset 84h) returns the masked version of the IntEvent register (IntEvent and IntMask).

Table 4-26. BAR0+Offset 80h (Set) and 84h (Clear): IntEvent Register

Bit	Name	Access	Reset	Description
31:27	RSVD	--	0	**Reserved**
26	phyRegRcvd	RSCU	Undef	**PHY Register Received:** The CS4210 has received a PHY register data byte which can be read from the PHY control register (see Section 4.4.20 "PHYControl Register" on page 77).
25	cycleTooLong	RSCU	Undef	**Cycle Too Long:** If LinkControl.cycleMaster (BAR0+Offset E0h[21] is set, this indicates that an isochronous cycle lasted longer than the allotted time. For implementations with a discrete cycleTooLong timer, hardware is expected to trigger this event no less than 115 seconds and no more than 120 seconds after sending a cycle start packet unless a subaction gap or bus reset indication is first observed. LinkControl.cycleMaster is cleared by this event.
24	unrecoverableError	RSCU	Undef	**Unrecoverable Error:** This event occurs when the CS4210 encounters any error that forces it to stop operations on any or all of its subunits. For example, when a DMA context sets its contextControl.dead bit. While unrecoverableError is set, all normal interrupts for the context(s) that caused this interrupt are blocked from being set.
23	cycleInconsistent	RSC	Undef	**Cycle Inconsistent:** A cycle start was received that had an isochronous cycleTimer.seconds and isochronous cycleTimer.count different from the value in the IsochCycleTimer register (BAR0+Offset F0h, see Section 4.4.21 "IsochCycleTimer Register" on page 77). The CS4210 indicates a cycleInconsistent if a host initiated write changes the cycleSeconds or cycleCount fields of the cycleTimer register. For the effect of this condition on isochronous transmit and receive, refer to Section 3.8.3 "Isochronous Transmit and Receive" on page 30.
22	cycleLost	RSCU	Undef	**Cycle Lost**: A lost cycle is indicated when no cycle_start packet is sent/ received between two successive cycleSynch events.
21	cycle64Seconds	RSCU	Undef	**Cycle 64 Seconds:** Indicates that the 7th bit of the cycle second counter has changed.

Register Descriptions (Continued)

Table 4-26. BAR0+Offset 80h (Set) and 84h (Clear): IntEvent Register (Continued)

Bit	Name	Access	Reset	Description
20	cycleSynch	RSCU	Undef	**Cycle Synchronous:** Indicates that a new isochronous cycle has started. Set when the low order bit of the internal IsochCycleTimer.cycleCount (BAR0+Offset F0h[24:12]) toggles.
19	phy	RSCU	Undef	**Physical Layer:** Generated when the CS4103 requests an interrupt through a status transfer.
18	RSVD	--	0	**Reserved**
17	busReset	RSCU	Undef	**Bus Reset:** Indicates that the CS4103 has entered bus reset mode. See Section 4.4.16.2 "Bus Reset" on page 72 for information on when to clear this interrupt.
16	selfIDcomplete	RSCU	Undef	**Self-ID Complete:** A Self-ID packet stream has been received. Is set at the end of the bus initialization process if LinkControl.rcvSelfID (BAR0+Offset E0h[9]) is set. This bit is turned off simultaneously when IntEvent.busReset (bit 17) is turned on.
15:10	RSVD	--	0	**Reserved**
9	lockRespErr	RSCU	Undef	**Lock Response Error:** Indicates that the CS4210 attempted to return a lock response for a lock request to a serial bus register described in Section 4.4.4 "Autonomous CSR Resources" on page 60, but did not receive an ack_complete after exhausting all permissible retries.
8	postedWriteErr	RSCU	Undef	**Posted Write Error:** Indicates that a host bus error occurred while the CS4210 was trying to write a 1394 write request, which had already been given an ack_complete, into system memory. The 1394 destination offset and sourceID are available in the PostedWriteAddress register described in Section 3.7.7 "Posted Write Error" on page 29.
7	isochRx	RU	Undef	**Isochronous Receive DMA interrupt:** Indicates that one or more isochronous receive contexts have generated an interrupt. This is not a latched event, it is the OR'ing all bits in (IsochRxIntEvent and IsochRxIntMask). The IsochRxIntEvent register indicates which contexts have interrupted. See Section 4.4.16.6 "IsochRxIntEvent Register" on page 74.
6	isochTx	RU	Undef	**Isochronous Transmit DMA interrupt:** Indicates that one or more isochronous transmit contexts have generated an interrupt. This is not a latched event, it is the OR'ing all bits in (isochTxIntEvent and isochTxIntMask). The isochTxIntEvent register indicates which contexts have interrupted. See Section 4.4.16.4 "IsochTxIntEvent Register" on page 73.
5	RSPkt	RSCU	Undef	**Receive Response Packet:** Indicates that a packet was sent to an asynchronous receive response context buffer and the descriptor's xferStatus and resCount fields have been updated. This differs from ARRS (bit 3) since RSPkt is a per-packet completion indication and ARRS is a per-command descriptor (buffer) completion indication. AR Response buffers may contain more than one packet.
4	RQPkt	RSCU	Undef	**Receive Request Packet:** Indicates that a packet was sent to an asynchronous receive request context buffer and the descriptor's xferStatus and resCount fields have been updated. This differs from ARRQ (bit 2) since RQPkt is a per-packet completion indication and ARRQ is a per-command descriptor (buffer) completion indication. AR Request buffers may contain more than one packet.
3	ARRS	RSCU	Undef	**Asynchronous Receive Response DMA Interrupt:** This bit is conditionally set upon completion of an ARDMA Response context command descriptor.
2	ARRQ	RSCU	Undef	**Asynchronous Receive Request DMA Interrupt:** This bit is conditionally set upon completion of an ARDMA Request context command descriptor.
1	respTxComplete	RSCU	Undef	**Asynchronous Response Transmit DMA Interrupt:** This bit is conditionally set upon completion of an ATDMA response OUTPUT_LAST* command.
0	reqTxComplete	RSCU	Undef	**Asynchronous Request Transmit DMA Interrupt:** This bit is conditionally set upon completion of an ATDMA request OUTPUT_LAST* command.

Register Descriptions (Continued)

4.4.16.2 Bus Reset

When a bus reset occurs and the busReset interrupt is set to one, the selfIDComplete (BAR0+Offset 80h[16]) interrupt is simultaneously cleared to 0. The CS4210 prevents software from clearing the busReset interrupt bit during the Self-ID phase of bus initialization. Software must take precautions regarding the asynchronous transmit contexts before clearing this interrupt. Refer to Section 3.8 "Bus Resets" on page 30 for further details.

4.4.16.3 IntMask Register

The bits in the IntMask register have the same format as the IntEvent register, with the addition of masterIntEnable (bit 31). A one bit in the IntMask register enables the corresponding IntEvent register bit to generate a processor interrupt. A zero bit in IntMask disables the corresponding IntEvent register bit from generating a processor interrupt.

A bit is set in the IntMask register by writing a one to the corresponding bit in the IntMask Set address and cleared by writing a one to the corresponding bit in the IntMask Clear address. If masterIntEnable is 0, all interrupts are disabled regardless of the values of all other bits in the IntMask register. The value of masterIntEnable has no effect on the value returned by reading the IntEvent Clear; even if masterIntEnable is 0, reading IntEvent Clear returns (IntEvent and IntMask) as described earlier in Section 4.4.16 "Interrupts" on page 70.

On a reset, the IntMask.masterIntEnable bit (31) is set to 0 and the value of all other bits is undefined.

1) BAR0+Offset 88h: IntMask Set
2) BAR0+Offset 8Ch: IntMask Clear

Table 4-27. BAR0+Offset 88h (Set) and 8Ch (Clear): IntMask Register

Bit	Name	Access	Reset	Description
31	masterIntEnable	RSC	0	**Master Interrupt Enable:** If set, external interrupts are generated in accordance with the IntMask register. If clear, no external interrupts are generated regardless of the IntMask register settings.
30:27	RSVD	--	0	
26	phyRegRcvdIntEn	RSC	Undef	**Interrupt Events:** A one bit enables the corresponding IntEvent register bit to generate a processor interrupt. A zero bit disables the corresponding IntEvent register bit from generating a processor interrupt. See Table 4-26 "BAR0+Offset 80h (Set) and 84h (Clear): IntEvent Register" on page 70.
25	cycleTooLongIntEn	RSC	Undef	
24	unrecoverableError-IntEn	RSC	Undef	
23	cycleInconsis-tentIntEn	RSC	Undef	
22	cycleLostIntEn	RSC	Undef	
21	cycle64SecondsIntEn	RSC	Undef	
20	cycleSynchIntEn	RSC	Undef	
19	phyIntEn	RSC	Undef	
18	RSVD	--	0	
17	busResetIntEn	RSC	Undef	
16	selfIDcompleteIntEn	RSC	Undef	
15:10	RSVD	--	0	
9	lockRespErrIntEn	RSC	Undef	
8	postedWriteErrIntEn	RSC	Undef	
7	isochRxIntEn	RSC	Undef	
6	isochTxIntEn	RSC	Undef	
5	RSPktIntEn	RSC	Undef	
4	RQPktIntEn	RSC	Undef	
3	ARRSIntEn	RSC	Undef	
2	ARRQIntEn	RSC	Undef	
1	respTxCompleteIntEn	RSC	Undef	
0	reqTxCompleteIntEn	RSC	Undef	

Register Descriptions (Continued)

4.4.16.4 IsochTxIntEvent Register

There are two 32-bit registers to report isochronous transmit context interrupts: IsochTxIntEvent and IsochTxIntMask. Both registers are set and clear (Tables 4-28 and 4-29). For all four addresses, writing a zero bit has no effect on the corresponding bit in the register.

The IsochTxIntEvent register contains the actual interrupt request bits. Each of these bits corresponds to a DMA completion event for the indicated isochronous transmit context. The IsochTxIntMask register is ANDed with the IsochTxIntEvent register to enable selected bits to generate processor interrupts. If IsochTxIntMask and IsochTxIntEvent are not zero, then the IntEvent.IsochTxIntn bit is set to one, and if enabled via the IntMask register it generates a processor interrupt. A software write to the IsochTxIntEvent Set register can therefore cause an interrupt (if not otherwise masked). A software write to the IsochTxIntEvent Clear register clears interrupt conditions reported in the IsochTxIntEvent register.

Reading the IsochTxIntEvent Set register returns the current state of the IsochTxIntEvent register. Reading the IsochTxIntEvent Clear register returns the masked version of the IsochTxIntEvent register (IsochTxIntEvent and IsochTxIntMask).

This IsochTxIntEvent register reflects the interrupt state of the isochronous transmit contexts. An interrupt is generated on behalf of an isochronous transmit context if an OUTPUT_LAST DMA command completes and its “i” field is set to 11b (interrupt always). Upon determining that the IntEvent.IsochTx interrupt has occurred, software can check the IsochTxIntEvent register to determine which context(s) caused the interrupt.

4.4.16.5 IsochTxIntMask Register

The bits in the IsochTxIntMask register (Table 4-29) have the same format as the IsochTxIntEvent register. Setting a bit in this register enables the corresponding bit in the IsochTxIntEvent register. Setting a bit in this register is done by setting the bit in the IsochTxIntMask Set register (BAR0+Offset 98h) and cleared by writing a one to the corresponding bit in the IsochTxIntMask Clear register (BAR0+Offset 9Ch). Bits for all unimplemented contexts read as 0’s. Software can use this register to determine which contexts are supported by writing to it with all 1’s, then reading it back. Contexts with a 1 are implemented, and those with a 0 are not.

Table 4-28. BAR0+Offset 90h (Set) and 94h (Clear): IsochTxIntEvent Register

Bit	Name	Access	Reset	Description
31:8	RSVD	--	0	**Reserved**
7	isochTxInt7	RSCU	Undef	**Isochronous Transmit Context Interrupt Event:** Set to one when the corresponding isochronous transmit context interrupts and the interrupt mask (BAR0+Offset 98h) is enabled.
6	isochTxInt6			
5	isochTxInt5			
4	isochTxInt4			
3	isochTxInt3			
2	isochTxInt2			
1	isochTxInt1			
0	isochTxInt0			

Table 4-29. BAR0+Offset 98h (Set) and 9Ch (Clear): IsochTxIntMask Register

Bit	Name	Access	Reset	Description
31:8	RSVD	--	0	**Reserved**
7	isochTxIntMask7	RSC	Undef	**Isochronous Transmit Context Interrupt Mask:** Set to one enables the corresponding bit in the IsochTxintEvent register (BAR0+Offset 90h).
6	isochTxIntMask6			
5	isochTxIntMask5			
4	isochTxIntMask4			
3	isochTxIntMask3			
2	isochTxIntMask2			
1	isochTxIntMask1			
0	isochTxIntMask0			

Register Descriptions (Continued)

4.4.16.6 IsochRxIntEvent Register

There are two 32-bit registers to report isochronous receive context interrupts: IsochRxIntEvent and IsochRxIntMask. Both registers are set and clear. For all four addresses, writing a “zero” bit has no effect on the corresponding bit in the register. The IsochRxIntEvent register contains the actual interrupt request bits. Each of these bits corresponds to a DMA completion event for the indicated isochronous receive context. The IsochRxIntMask register is ANDed with the IsochRxIntEvent register to enable selected bits to generate processor interrupts. If (IsochRxIntMask and IsochRxIntEvent) are not zero, then the IsochRxIntn bit is set to one, and if enabled via the IntMask register it generates a processor interrupt. A software write to the IsochRxIntEvent Set register can therefore cause an interrupt (if not otherwise masked). A software write to the IsochRxIntEvent Clear register clears interrupt conditions reported in the IsochRxIntEvent register. Reading the IsochRxIntEvent Set register returns the current state of the IsochRxIntEvent register. Reading the IsochRxIntEvent Clear register returns the masked version of the IsochRxIntEvent register (IsochRxIntEvent and IsochRxIntMask). The IsochRxIntEvent register reflects the interrupt state of the isochronous receive contexts. An interrupt is generated on behalf of an isochronous receive context if a final command of a DMA descriptor block completes and its i bits are set to 11b (interrupt always). Upon determining that the IsochRx interrupt has occurred, software can check the IsochRxIntEvent register to determine which context(s) caused the interrupt.

4.4.16.7 IsochRxIntMask Register

The bits in the IsochRxIntMask register have the same format as the IsochRxIntEvent register. Setting a bit in this register enables the corresponding bit in the IsochRxIntMask Set register and is cleared by writing a one to the corresponding bit in the IsochRxIntMask Clear register. Bits for all unimplemented contexts read as 0’s. Software can use this register to determine which contexts are supported by writing to it with all 1’s then reading it back. Contexts with a 1 are implemented, and those with a 0 are not.

Table 4-30. BAR0+Offset A0h (Set) and A4h (Clear): IsochRxIntEvent Register

Bit	Name	Access	Reset	Description
31:8	RSVD	--	0	**Reserved**
7	isochRxInt7	RSC	Undef	**Isochronous Receive Contexts Interrupt Event:** Set to one when the corresponding isochronous receive context interrupts and the Interrupt Mask is enabled.
6	isochRxInt6			
5	isochRxInt5			
4	isochRxInt4			
3	isochRxInt3			
2	isochRxInt2			
1	isochRxInt1			
0	isochRxInt0			

Table 4-31. BAR0+Offset A8h (Set) and ACh (Clear): IsochRxIntMask Register

Bit	Name	Access	Reset	Description
31:8	RSVD	--	0	**Reserved**
7	isochRxIntMask7	RSC	Undef	**Isochronous Receive Contexts Interrupt Mask Set:** Set to one enables the corresponding bit in the IsochRxIntEvent Register.
6	isochRxIntMask6			
5	isochRxIntMask5			
4	isochRxIntMask4			
3	isochRxIntMask3			
2	isochRxIntMask2			
1	isochRxIntMask1			
0	isochRxIntMask0			

Register Descriptions (Continued)

4.4.17 Fairness Control Register

This register (Table 4-32) provides a mechanism by which software can direct the CS4210 to transmit multiple asynchronous request packets during a fairness interval as specified in P1394a specification.

4.4.18 LinkControl Register

This register (Table 4-33) provides the control flags that enable and configure the link core protocol and controls for the receiver and cycle timer. This register is set (BAR0+Offset E0h) and clear (BAR0+Offset E4h). On read, both addresses return LinkControl.

Table 4-32. BAR0+Offset DCh: Fairness Control Register

Bit	Name	Access	Reset	Description
31:8	RSVD	--	0	**Reserved**
7:0	pri_req	RW	HW = Undef SW/Bus = N/A	**Priority Arbitration Request:** This field specifies the maximum number of priority arbitration requests for asynchronous request packets that the link is permitted to make to the CS4103 during a fairness interval. A pri_req value of 00h is equivalent to the behavior specified by the IEEE 1394-1995 specification.

Table 4-33. BAR0+Offset E0h (Set) and E4h (Clear): LinkControl Register

Bit	Name	Access	Reset	Description
32:21	RSVD	--	0	**Reserved**
22	cycleSource	RSC or R	HW = 0 SW = No Change	**Cycle Source:** When one, the cycle timer uses an external source to determine when to increment cycleCount (BAR0+Offset F0h[24:12]). When cycleCount is incremented, cycleOffset (BAR0+Offset F0h[11:0]) is reset to 0. If cycleOffset reaches 3071 before an external event occurs, it remains at 3071 until the external signal is received and is then reset to 0. When the cycleSource bit is zero, the CS4210 rolls the cycle timer over when the timer reaches 3072 cycles of the 24.576 MHz clock (8 kHz). CycleSource has an effect only when cycleMaster (bit 21) is enabled. A hardware reset clears to 0. A software reset has no effect.
21	cycleMaster	RSCU	Undef	**Cycle Master:** When one and the CS4103 has notified the CS4210 that it is root, the CS4210 generates a cycle start packet every time the cycle timer rolls over, based on the setting of the cycleSource bit (bit 22). When zero, the CS4210 accepts received cycle start packets to maintain synchronization with the node which is sending them. This bit is automatically zeroed when the IntEvent.cycleTooLong event occurs and cannot be set until the IntEvent.cycleTooLong bit (BAR0+Offset 80h[25]) is cleared.
20	cycleTimerEnable	RSC	Undef	**Cycle Timer Enable:** When one, the cycle timer offset counts cycles of the 24.576 MHz clock and rolls over at the appropriate time based on the settings of the above bits. When zero, the cycle timer offset will not count.
19:11	RSVD	--	0	**Reserved**
10	rcvPhyPkt	RSC	Undef	**Receive Physical Layer Packet:** When one, the receiver accepts incoming CS4103 packets into the AR request context if the AR request context is enabled. This does not control either the receipt of self-identification packets during the Self-ID phase of bus initialization or the queuing of synthesized bus reset packets in the ARDMA Request Context buffer (see Section 3.8 "Bus Resets" on page 30). This does control receipt of any self- identification packets received outside of the Self-ID phase of bus initialization.
9	rcvSelfID	RSC	Undef	**Receive Self-ID:** When one, the receiver accepts incoming self-identification packets. Before setting this bit to one, software must ensure that the selfIDBufferPtr register bits (BAR0+Offset 64h[31:11]) contains a valid address.
8:0	RSVD	--	0	**Reserved**

Register Descriptions (Continued)

4.4.19 Node ID and Status Register

This register contains the CSR address for the node on which this chip resides. The 16-bit combination of busNumber and nodeNumber is referred to as the Node ID. This register is written autonomously and atomically by the CS4210 with the value in CS4103's base register at Address 00h following the self-identification phase of bus initialization. Although IntEvent.phyRegRcvd (BAR0+Offset 80h[26]) is not set when the contents of CS4103's base register at Address 00h are written here, software can use the IntEvent.selfIDComplete (BAR0+Offset 80h[16]) interrupt to detect that the self-identification phase has completed and can then check for a new valid Node ID.

Table 4-34. BAR0+Offset E8h: Node ID and Status Register

Bit	Name	Access	Reset	Description
31	iDValid	RU	0	**ID Valid:** This bit indicates whether or not the CS4210 has a valid node number. It is cleared when the bus reset state is detected and set again when the CS4210 receives a new node number from the CS4103. If iDValid is clear, software should not set ContextControl.run for either of the ATDMA contexts (request and response).
30	root	RU	0	**Root:** This bit is set during the bus reset process if the CS4103 is root.
29:28	RSVD	--	0	**Reserved**
27	CPS	RU	0	**Cable Power Status:** Set if the CS4103 is reporting that cable power status is OK (VP 8V).
26:16	RSVD	--	0	**Reserved**
15:6	busNumber	RWU	3FFh	**Bus Number**: This number is used to identify the specific 1394 bus this node belongs to when multiple 1394-compatible buses are connected via a bridge. This field is set to 3FFh on a bus reset.
5:0	nodeNumber	RU	Undef	**Node Number:** This number is the physical node number established by the CS4103 during self-identification. It is automatically set to the value received from the CS4103 after the self-identification phase. If the CS4103 sets the nodeNumber to 63, software should not set ContextControl.run for either of the ATDMA contexts (request and response).

Register Descriptions (Continued)

4.4.20 PHYControl Register

The PHYControl register (Table 4-35) is used to read or write a CS4103 register. To read a register, the address of the register is written to the regAddr field along with a 1 in the rdReg bit. When the read request has been sent to the CS4103 (through the LREQ pin), the rdReg bit is cleared to 0. When the CS4103 returns the register, the rdDone bit transitions to 1 and the IntEvent.phyRegRcvd interrupt (BAR0+Offset 80h[26]) is set. The address of the register received is placed in the rdAddr field and the contents in the rdData field. Software must not issue a read of CS4103 register 0. The most recently available contents of this register is reflected in the NodeID register (see Section 4.4.19 "Node ID and Status Register" on page 76). To write to a CS4103 register, the address of the register is written to the regAddr field, the value to write to the wrData field, and a 1 to the wrReg bit. The wrReg bit is cleared when the write request has been transferred to the CS4103. Software must serialize all CS4103 register reads and writes. Only after the current CS4103 register read or write completes may software issue a different CS4103 register read or write.

4.4.21 IsochCycleTimer Register

The IsochCycleTimer register (Table 4-36) is a read/write register that shows the current cycle number and offset. The cycle timer register is split up into three fields. The lower order 12 bits are the cycle offset, the middle 13 bits are the cycle count, and the upper order 7 bits count time in seconds. When the CS4210 is cycle master, this register is transmitted with the cycle start message. When the CS4210 is not the cycle master, this register is loaded with the data field in each incoming cycle start. In the event that the cycle start message is not received, the fields continue incrementing on their own (when cycleTimerEnable is set in the LinkControl register, BAR0+Offset E0h[20]) to maintain a local time reference.

Table 4-35. BAR0+Offset ECh: PHYControl Register

Bit	Name	Access	Reset	Description
31	rdDone	RU	Undef	**Read Done:** rdDone is cleared to 0 by the CS4210 when either rdReg or wrReg is set to 1. This bit is set to 1 when a register transfer is received from the CS4103.
30:28	RSVD	--	0	**Reserved**
27:24	rdAddr	RU	Undef	**Read Address:** This is the address of the register most recently received from the CS4103.
23:16	rdData	RU	Undef	**Read Data:** Contains the data read from the CS4103 register at rdAddr.
15	rdReg	RWU	0	**Read Register:** Set rdReg to initiate a read request to a CS4103 register. This bit is cleared when the read request has been sent. The wrReg bit must not be set while the rdReg bit is set.
14	wrReg	RWU	0	**Write Register:** Set wrReg to initiate a write request to a CS4103 register. This bit is cleared when the write request has been sent. The rdReg bit must not be set while the wrReg bit is set.
13:12	RSVD	--	0	**Reserved**
11:8	regAddr	RW	Undef	**Register Address:** regAddr is the address of the CS4103 register to be written or read.
7:0	wrData	RWU	Undef	**Write Data:** This is the contents to be written to a CS4103 register. Ignored for a read.

Table 4-36. BAR0+Offset F0h: IsochCycleTimer Register

Bit	Name	Access	Reset	Description
31:25	cycleSeconds	RWU	N/A	**Cycle Seconds:** This field counts seconds (cycleCount rollovers) modulo 128.
24:12	cycleCount	RWU	N/A	**Cycle Count:** This field counts cycles (cycleOffset rollovers) modulo 8000.
11:0	cycleOffset	RWU	N/A	**Cycle Offset**: This field counts 24.576 MHz clocks modulo 3072, (i.e., 125 µs). If an external 8 kHz clock configuration is being used, cycleOffset is set to 0 at each tick of the external clock.

Register Descriptions (Continued)

4.4.22 Asynchronous Request Filter Registers

The CS4210 allows for selective access to host memory and the Asynchronous Receive Request context so that software can maintain host memory integrity. The selective access is provided by two sets of registers: PhysRequestFilter and AsyncRequestFilter. These registers allow access to physical memory and the AR Request context on a nodeID basis. The request filters are not applied to quadlet read requests directed at the Config ROM (including the ConfigROM header, BusID, Bus Options, and Global Unique ID registers) nor to accesses directed to the isochronous resource management registers. When the link is enabled, access by any node to the first 1K of CSR config ROM is enabled (see Section 4.4.5 "Configuration ROM Header Register" on page 61). The Asynchronous Request Filters do not have any effect on Asynchronous Response packets. When a request is received by the CS4210 from the 1394 bus and that request does not access the first 1K of CSR config ROM on the CS4210, then the sourceID is used to index into the AsyncRequestFilter. If the corresponding bit in the AsyncRequestFilter is set to 0, then requests from that device are not enabled; there is no ack_sent, and the requests are ignored by the CS4210. If, however, the bit is set to 1, the requests are accepted and processed according to the address of the request and the setting of the PhysicalRequestFilter register (BAR0+Offset 110h, see Section 4.4.23 "Physical Request Filter Registers" on page 79). Requests to offsets above 0000_FFFF_FFFFh, with the exception of offsets handled physically as described in Section 3.6 "Physical Requests" on page 26, are always sent to the Asynchronous Request Receive DMA context. If the AR Request DMA context is not enabled, then the CS4210 ignores the request. These registers are set and clear. If bit asyncReqResourceN is set, then requests with a sourceID of either {3FFh, #N} or {busID, #N} are accepted. If the asyncReqResourceAll bit is set in AsyncRequestFilterHi, requests from all bus nodes including those on the local bus are accepted. Reading the AsyncRequestFilter registers returns their current state. All asyncReqResourceN bits in the AsyncRequestFilter register are cleared to 0 on a 1394 bus reset.

Table 4-37. BAR0+Offset 100h (Set) and 104h (Clear): AsyncRequestFilterHi Register

Bit	Name	Access	Reset	Description
31	asyncReqResourceAll	RSCU	0	**Asynchronous Requests Resource All:** If set to one, all asynchronous requests received by the CS4210 from all bus nodes (including the local bus) are accepted, and the values of all asyncReqResourceN bits are ignored. A bus reset does not affect the value of the asyncReqResourceAll bit.
30:0	asyncReqResourceN	RSCU	00h	**Asynchronous Requests Resource [62:32]:** If set to one for local bus node number N, asynchronous requests received by the CS4210 from that node are accepted. All asyncReqResourceN bits are cleared to zero when a bus reset occurs.

Table 4-38. BAR0+Offset 108h (Set) and 10Ch (Clear): AsyncRequestFilterLo Register

Bit	Name	Access	Reset	Description
31:0	asyncReqResourceN	RSCU	00h	**Asynchronous Requests Resource [31:0]:** If set to one for local bus node number N, asynchronous requests received by the CS4210 from that node are accepted. All asyncReqResourceN bits are cleared to zero when a bus reset occurs.

Register Descriptions (Continued)

4.4.23 Physical Request Filter Registers

If an asynchronous request is allowed from a node, and the offset is 0000_FFFF_FFFFh, the sourceID of the request is used as an index into the PhysicalRequestFilter. If the corresponding bit in the PhysicalRequestFilter is set to 0, then the request is forwarded to the Asynchronous Request Receive DMA context. If, however, the bit is set to 1, then the request is sent to the physical response unit. (Note that within the physical range, lock transactions and block transactions with a non-zero extended tcode are always forwarded to the Asynchronous Request Receive DMA context (see Section 3.6 "Physical Requests" on page 26.). This register is set and clear. If bit physReqResourceN is set, then requests with a sourceID of either {3FFh, #n} or {busID, #n} (where n is the node number) are accepted. If the physReqResourceAllBuses bit is set in PhysicalRequestFilterHi, physical requests from any device on any other bus are accepted (bus number other than 3FFh and busID). Physical requests that are rejected by the PhysicalRequestFilter are sent to the AR Request DMA context if the AR Request DMA context is enabled. If it is disabled, then the CS4210 ignores the requests. Reading the PhysicalRequestFilter registers returns their current state. All bits in the PhysicalRequestFilter are set to 0 on a 1394 bus reset.

Table 4-39. BAR0+Offset 110h (Set) and 114h (Clear): PhysicalRequestHi Register

Bit	Name	Access	Reset	Description
31	physReqResourceAllBuses	RSCU	0	**Asynchronous Physical Requests Resource All Buses:** If set to one, all asynchronous physical requests received by the CS4210 from non-local bus nodes are accepted.
30:0	physReqResourceN	RSCU	00h	**Asynchronous Physical Requests Resource [62:32]:** If set to one for local bus node number N, then asynchronous physical requests received by the CS4210 from that node are accepted.

Table 4-40. BAR0+Offset 118h (Set) and 11Ch (Clear): PhysicalRequestLo Register

Bit	Name	Access	Reset	Description
31:0	physReqResourceN	RSCU	00h	**Asynchronous Physical Requests Resource [31:0]:** If set to one for local bus node number N, then asynchronous physical requests received by the CS4210 from that node are accepted.

Register Descriptions (Continued)

4.4.24 Asynchronous Request/Response Transmit

Each ATDMA context (request and response) has two registers: CommandPtr and ContextControl. CommandPtr is used by software to tell the CS4210 where the DMA context program begins. ContextControl is used by software to control the context's behavior, and is used by hardware to indicate current status.

4.4.24.1 Async Request Transmit Context Control Register

The ContextControl Set/Clear registers contain bits that control options, operational state and status for the DMA context. Software can set selected bits by writing ones to the corresponding bits in the ContextControl Set register. Software can clear selected bits by writing ones to the corresponding bits in the ContextControl Clear register. It is not possible for software to set some bits and clear others in an atomic operation. A read from either register returns the same value.

4.4.24.2 Async Request Transmit Command Pointer Register

Software initializes CommandPtr.descriptorAddress to contain the address of the first descriptor block that the CS4210 accesses when software enables the context by setting ContextControl.run. Software also initializes CommandPtr.Z to indicate the number of descriptors in the first descriptor block. Software only writes to this register when both ContextControl.run and ContextControl.active are zero.

Table 4-41. BAR0+Offset 180h (Set) and 184h (Clear): AsyncReqTxContextControl Register

Bit	Name	Access	Reset	Description
31:16	RSVD	--	0	**Reserved**
15	run	RSCU	0	**Run:** The run bit is set by software to enable descriptor processing for a context and cleared by software to stop descriptor processing. The CS4210 only clears this bit on a hardware or software reset. See Section 3.3.2.1 "ContextControl.run" on page 21 for details.
14:13	RSVD	--	0	**Reserved**
12	wake	RSU	Undef	**Wake:** Software sets this bit to 1 to cause the CS4210 to continue or resume descriptor processing. The CS4210 clears this bit on every descriptor fetch. See Section 3.3.2.2 "ContextControl.wake" on page 21 for details.
11	dead	RU	0	**Dead:** The CS4210 sets this bit when it encounters a fatal error. The CS4210 clears this bit when software clears the run bit. See Section 3.3.2.4 "ContextControl.dead" on page 21 for details.
10	active	RU	0	**Active**: The CS4210 sets this bit to 1 when it is processing descriptors. See Section 3.3.2.3 "ContextControl.active" on page 21 for details.
9:5	RSVD	--	0	**Reserved**
4:0	event code	RU	Undef	**Event Code:** Following an OUTPUT_LAST* command, the received ack_code or an "evt_" error code is indicated in this field. Possible values are: ack_complete, ack_pending, ack_busy_X, ack_busy_A, ack_busy_B, ack_data_error, ack_type_error, evt_tcode_err, evt_missing_ack, evt_underrun, evt_descriptor_read, evt_data_read,evt_timeout, evt_flushed, and evt_unknown.

Table 4-42. BAR0+Offset 18Ch: AsyncReqTxCommandPtr Register

Bit	Name	Access	Reset	Description
31:4	descriptorAddress	RWU	Undef	**Descriptor Address:** Contains the upper 28 bits of the address of a 16-byte aligned descriptor block. See Section 3.3.2.5 "CommandPtr" on page 22 for details.
3:0	Z	RWU	Undef	**Z Bit:** Indicates the number of contiguous 16-byte aligned blocks at the address pointed to by descriptorAddress. If Z is 0, it indicates that the descriptorAddress is not valid.

Register Descriptions (Continued)

4.4.24.3 Async Response Transmit Context Control Register

The ContextControl Set/Clear registers contain bits that control options, operational state and status for the DMA context. Software can set selected bits by writing ones to the corresponding bits in the ContextControl Set register. Software can clear selected bits by writing ones to the corresponding bits in the ContextControl Clear register. It is not possible for software to set some bits and clear others in an atomic operation. A read from either register returns the same value.

4.4.24.4 Async Response Transmit Command Pointer Register

Software initializes CommandPtr.descriptorAddress to contain the address of the first descriptor block that the CS4210 accesses when software enables the context by setting ContextControl.run. Software also initializes CommandPtr.Z to indicate the number of descriptors in the first descriptor block. Software only writes to this register when both ContextControl.run and ContextControl.active are zero.

Table 4-43. BAR0+Offset 1A0h (Set) and 1A4h (Clear): AsyncRespTxContextControl Register

Bit	Name	Access	Reset	Description
31:16	RSVD	--	0	**Reserved**
15	run	RSCU	0	**Run:** The run bit is set by software to enable descriptor processing for a context and cleared by software to stop descriptor processing. The CS4210 only clears this bit on a hardware or software reset. See Section 3.3.2.1 "ContextControl.run" on page 21 for details.
14:13	RSVD	--	0	**Reserved**
12	wake	RSU	Undef	**Wake:** Software sets this bit to 1 to cause the CS4210 to continue or resume descriptor processing. The CS4210 clears this bit on every descriptor fetch. See Section 3.3.2.2 "ContextControl.wake" on page 21 for details.
11	dead	RU	0	**Dead:** The CS4210 sets this bit when it encounters a fatal error. The CS4210 clears this bit when software clears the run bit. See Section 3.3.2.4 "ContextControl.dead" on page 21 for details.
10	active	RU	0	**Active:** The CS4210 sets this bit to 1 when it is processing descriptors. See Section 3.3.2.3 "ContextControl.active" on page 21 for details.
9:5	RSVD	--	0	**Reserved**
4:0	eventcode	RU	Undef	**Event Code:** Following an OUTPUT_LAST* command, the received ack_code or an “evt_” error code is indicated in this field. Possible values are: ack_complete, ack_pending, ack_busy_X, ack_busy_A, ack_busy_B, ack_data_error, ack_type_error, evt_tcode_err, evt_missing_ack, evt_underrun, evt_descriptor_read, evt_data_read, evt_timeout, evt_flushed, and evt_unknown.

Table 4-44. BAR0+Offset 1ACh: AsyncRespTxCommandPtr Register

Bit	Name	Access	Reset	Description
31:4	descriptorAddress	RWU	Undef	**Descriptor Address:** Contains the upper 28 bits of the address of a 16-byte aligned descriptor block. See Section 3.3.2.5 "CommandPtr" on page 22 for details.
3:0	Z	RWU	Undef	**Z Bit:** Indicates the number of contiguous 16-byte aligned blocks at the address pointed to by descriptorAddress. If Z is 0, it indicates that the descriptorAddress is not valid.

Register Descriptions (Continued)

4.4.25 Asynchronous Request/Response Receive

Each ARDMA receive context (request and response) has a CommandPtr register and a ContextControl register. CommandPtr is used by software to tell the CS4210 where the DMA context program begins. ContextControl is used by software to control the context's behavior, and is used by hardware to indicate current status.

4.4.25.1 Async Request Receive Context Control Register

The ContextControl Set/Clear registers contain bits that control options, operational state, and status for the DMA context. Software can set selected bits by writing ones to the corresponding bits in the ContextControl Set register. Software can clear selected bits by writing ones to the corresponding bits in the ContextControl Clear register. It is not possible for software to set some bits and clear others in an atomic operation. A read from either register returns the same.

4.4.25.2 Async Request Receive Command Pointer Register

The CommandPtr register specifies the address of the context program to be executed when a DMA context is started. All descriptors are 16-byte aligned, so the four least-significant bits of any descriptor address must be zero. The least-significant bit of the CommandPtr register is used to encode a Z value. For each ARDMA context (request and response) Z may be either 1 to indicate that descriptorAddress points to a valid command descriptor, or 0 to indicate that there are no descriptors in the context program.

Table 4-45. BAR0+Offset 1C0h (Set) and 1C4h (Clear): AsyncReqRxContextControl Register

Bit	Name	Access	Reset	Description
31:16	RSVD	--	0	**Reserved**
15	run	RSCU	0	**Run:** The run bit is set by software to enable descriptor processing for a context and cleared by software to stop descriptor processing. The CS4210 only clears this bit on a hardware or software reset. See Section 3.3.2.1 "ContextControl.run" on page 21 for details.
14:13	RSVD	--	0	**Reserved**
12	wake	RSU	Undef	**Wake:** Software sets this bit to 1 to cause the CS4210 to continue or resume descriptor processing. The CS4210 clears this bit on every descriptor fetch. See Section 3.3.2.2 "ContextControl.wake" on page 21 for details.
11	dead	RU	0	**Dead:** The CS4210 sets this bit when it encounters a fatal error. The CS4210 clears this bit when software clears the run bit. See Section 3.3.2.4 "ContextControl.dead" on page 21 for details.
10	active	RU	0	**Active:** The CS4210 sets this bit to 1 when it is processing descriptors. See Section 3.3.2.3 "ContextControl.active" on page 21 for details.
9:8	RSVD	--	0	**Reserved**
7:5	spd	RU	Undef	**Speed:** This field indicates the speed at which the last packet was received by this context. 000 = 100 Mbits/sec, 001 = 200 Mbits/sec and 010 = 400 Mbits/sec. All other values are reserved. Software should not attempt to interpret the contents of this field while the ContextControl.active or ContextControl.wake bits are set.
4:0	eventcode	RU	undef	**Event Code:** The packet ack_ code or an "evt_" error code is indicated in this field. Possible values are: ack_complete, ack_pending, ack_type_error, evt_descriptor_read, evt_data_write, evt_bus_reset, evt_unknown, and evt_no_status.

Table 4-46. BAR0+Offset 1CCh: AsyncReqRxCommandPtr Register

Bit	Name	Access	Reset	Description
31:4	descriptorAddress	RWU	Undef	**Descriptor Address:** Contains the upper 28 bits of the address of a 16-byte aligned descriptor block. See Section 3.3.2.5 "CommandPtr" on page 22 for details.
3:0	Z	RWU	Undef	**Z Bit:** May be either 1 to indicate that descriptorAddress points to a valid command descriptor, or 0 to indicate that there are no descriptors in the context program.

Register Descriptions (Continued)

4.4.25.3 Async Response Receive Context Control Register

The ContextControl Set/Clear registers contain bits that control options, operational state, and status for the DMA context. Software can set selected bits by writing ones to the corresponding bits in the ContextControl Set register. Software can clear selected bits by writing ones to the corresponding bits in the ContextControl Clear register. It is not possible for software to set some bits and clear others in an atomic operation. A read from either register returns the same.

4.4.25.4 Async Response Receive Command Pointer Register

The CommandPtr register specifies the address of the context program which is executed when a DMA context is started. All descriptors are 16-byte aligned, so the four least-significant bits of any descriptor address must be zero. The least-significant bit of the CommandPtr register is used to encode a Z value. For each ARDMA context (request and response) Z may be either 1 to indicate that descriptorAddress points to a valid command descriptor, or 0 to indicate that there are no descriptors in the context program.

Table 4-47. BAR0+Offset 1E0h (Set) and 1E4h (Clear): AsyncRespRxContextControl Register

Bit	Name	Access	Reset	Description
31:16	RSVD	--	0	**Reserved**
15	run	RSCU	0	**Run:** The run bit is set by software to enable descriptor processing for a context and cleared by software to stop descriptor processing. The CS4210 only clears this bit on a hardware or software reset. See Section 3.3.2.1 "ContextControl.run" on page 21 for details.
14:13	RSVD	--	0	**Reserved**
12	wake	RSU	Undef	**Wake:** Software sets this bit to 1 to cause the CS4210 to continue or resume descriptor processing. The CS4210 clears this bit on every descriptor fetch. See Section 3.3.2.2 "ContextControl.wake" on page 21 for details.
11	dead	RU	0	**Dead:** The CS4210 sets this bit when it encounters a fatal error. The CS4210 clears this bit when software clears the run bit. See Section 3.3.2.4 "ContextControl.dead" on page 21 for details.
10	active	RU	0	**Active:** The CS4210 sets this bit to 1 when it is processing descriptors. See Section 3.3.2.3 "ContextControl.active" on page 21 for details.
9:8	RSVD	--	0	**Reserved**
7:5	spd	RU	Undef	**Speed:** This field indicates the speed at which the last packet was received by this context. 000 = 100 Mbits/sec, 001 = 200 Mbits/sec and 010 = 400 Mbits/sec. All other values are reserved. Software should not attempt to interpret the contents of this field while the ContextControl.active or ContextControl.wake bits are set.
4:0	eventcode	RU	Undef	**Event Code:** The packet ack_ code or an “evt_” error code is indicated in this field. Possible values are: ack_complete, ack_pending, ack_type_error, evt_descriptor_read, evt_data_write, evt_bus_reset, evt_unknown, and evt_no_status.

Table 4-48. BAR0+Offset 1ECh: AsyncRespRxCommandPtr Register

Bit	Name	Access	Reset	Description
31:4	descriptorAddress	RWU	Undef	**Descriptor Address:** Contains the upper 28 bits of the address of a 16-byte aligned descriptor block. See Section 3.3.2.5 "CommandPtr" on page 22 for details.
3:0	Z	RWU	Undef	**Z Bit:** May be either 1 to indicate that descriptorAddress points to a valid command descriptor, or 0 to indicate that there are no descriptors in the context program.

Register Descriptions (Continued)

4.4.26 Isochronous Transmit

Each isochronous transmit context consists of two registers: CommandPtr and ContextControl. CommandPtr is used by software to tell the ITDMA controller where the DMA context program begins. IsochTxContextControl is used by software to control the context's behavior, and is used by hardware to indicate current status.

The CS4210 has eight isochronous transmit contexts. These registers are repeated at offsets of 10h times the context number. Table 4-49 is a map providing the offset addresses for the isochronous transmit ContextControl and CommandPtr registers.

Table 4-49. IsochTx Register Address Map

BAR0+Offset	Name
200h	IsochTx0ContextControl Set Register
204h	IsochTx0ContextControl Clear Register
208h	Reserved
20Ch	IsochTx0CommandPtr Register
210h	IsochTx1ContextControl Set Register
214h	IsochTx1ContextControl Clear Register
218h	Reserved
21Ch	IsochTx1CommandPtr Register
220h	IsochTx2ContextControl Set Register
224h	IsochTx2ContextControl Clear Register
228h	Reserved
22Ch	IsochTx2CommandPtr Register
230h	IsochTx3ContextControl Set Register
234h	IsochTx3ContextControl Clear Register
238h	Reserved
23Ch	IsochTx3CommandPtr Register
240h	IsochTx4ContextControl Set Register
244h	IsochTx4ContextControl Clear Register
248h	Reserved
24Ch	IsochTx4CommandPtr Register
250h	IsochTx5ContextControl Set Register
254h	IsochTx5ContextControl Clear Register
258h	Reserved
25Ch	IsochTx5CommandPtr Register
260h	IsochTx6ContextControl Set Register
264h	IsochTx6ContextControl Clear Register
268h	Reserved
26Ch	IsochTx6CommandPtr Register
270h	IsochTx7ContextControl Set Register
274h	IsochTx7ContextControl Clear Register
278h	Reserved
27Ch	IsochTx7CommandPtr Register

4.4.26.1 Isoch Transmit Context Control Register

The IsochTxContextControl Set/Clear registers (Table 4-50 on page 85) contain bits that control options, operational state, and status for the ITDMA contexts. Software can set selected bits by writing ones to the corresponding bits in the ContextControl Set register. Software can clear selected bits by writing ones to the corresponding bits in the ContextControl Clear register. It is not possible for software to set some bits and clear others in an atomic operation. A read from either register returns the same value.

In addition to the standard ContextControl fields, it includes a mechanism for starting transmit at a specified cycle time.

The cycleMatch field is used to start an ITDMA context program on a specified cycle. Software enables matching by setting the cycleMatchEnable bit. When the low order two bits of the bus IsochCycleTimer.cycleSeconds and IsochCycleTimer.cycleCount (BAR0+Offset F0h) value matches the cycleMatch value, hardware clears the cycleMatchEnable bit to 0, sets the ContextControl.active bit to 1, and begins executing descriptor blocks for the context. The transition of an ITDMA context to the active state from the not-active state is dependent upon the values of the run and cycleMatchEnable bits.

If run transitions to 1 when cycleMatchEnable is 0, then the context becomes active (active = 1).

If both run and cycleMatchEnable are set to 1, then the context becomes active when the low order two bits of the bus IsochCycleTimer.cycleSeconds and 13-bit IsochCycleTimer. cycleCount values match the 15-bit cycleMatch value.

If both run and cycleMatchEnable are set to 1, and cycleMatchEnable is subsequently cleared, the context becomes active.

If both run and active are 1 (the context is active), and then cycleMatchEnable is set to 1, this results in unspecified behavior.

Due to software latencies, software attempts to manage the startup of a context too close to the current time may not be effective.

In addition, the usability of cycleMatchEnable for IT contexts is impacted by the cycleInconsistent interrupt. Refer to Section 3.2.3.3 "Isoch Tx and Rx Context Interrupts" on page 17 for more information.

Register Descriptions (Continued)

Table 4-50. IsochTxnContextControl Set/Clear Register

Bit	Name	Access	Reset	Description
31	cycleMatchEnable	RSCU	Undef	**Cycle Match Enable:** When set to one, processing occurs such that the packet described by the context's first descriptor block is transmitted in the cycle whose number is specified in the cycleMatch field of this register. The 15-bit cycleMatch field must match the low order two bits of cycleSeconds and the 13-bit cycleCount field in the cycle start packet that is sent or received immediately before isochronous transmission begins. Since the ITDMA controller may work ahead, the processing of the first descriptor block may begin slightly in advance of the actual cycle in which the first packet is transmitted. The effects of this bit are impacted by the values of other bits in this register and are explained in descriptions below. Once the context has become active, hardware clears the cycleMatchEnable bit.
30:16	cycleMatch	RSC	Undef	**Cycle Match:** Contains a 15-bit value, corresponding to the low order two bits of the bus IsochCycleTimer.cycleSeconds and the 13-bit IsochCycleTimer.cycleCount field (BAR0+Offset F0h). If ContextControl.cycleMatchEnable is set, then this ITDMA context becomes enabled for transmits when the low order two bits of the bus IsochCycleTimer.cycleSeconds and IsochCycleTimer.cycleCount value equals the cycleMatch value.
15	run	RSCU	0	**Run:** The run bit is set by software to enable descriptor processing for a context and cleared by software to stop descriptor processing. The CS4210 only changes this bit on a hardware or software reset to set it to 0. See Section 3.3.2.1 "ContextControl.run" on page 21 for details.
14:13	RSVD	--	0	**Reserved**
12	wake	RSU	Undef	**Wake:** Software sets this bit to 1 to cause the CS4210 to continue or resume descriptor processing. The CS4210 clears this bit on every descriptor fetch. See Section 3.3.2.2 "ContextControl.wake" on page 21 for details.
11	dead	RU	0	**Dead:** The CS4210 sets this bit when it encounters a fatal error and clears this bit when software clears the run bit. See Section 3.3.2.4 "ContextControl.dead" on page 21 for details.
10	active	RU	0	**Active:** The CS4210 sets this bit to 1 when it is processing descriptors. See Section 3.3.2.3 "ContextControl.active" on page 21 for details.
9:5	RSVD	--	0	**Reserved**
4:0	event code	RU	Undef	**Event Code:** Following an OUTPUT_LAST* command, the error code is indicated in this field. Possible values are: ack_complete, evt_underrun, evt_descriptor_read, evt_data_read, evt_tcode_err and evt_unknown.

4.4.26.2 Isoch Transmit Command Pointer

The CommandPtr register (Table 4-51) specifies the address of the context program which is executed when a ITDMA context is started. All descriptors are 16-byte aligned, so the four least-significant bits of any descriptor address must be zero. The four least-significant bits of the CommandPtr register are used to encode a Z value that indicates how many physically contiguous descriptors are pointed to by descriptorAddress.

These registers are repeated at offsets of 10h times the context number (see Table 4-49 on page 84 for offset address assignment).

Table 4-51. IsochTxnCommandPtr Register

Bit	Name	Access	Reset	Description
31:4	descriptorAddress	RWU	Undef	**Descriptor Address:** Contains the upper 28 bits of the address of a 16-byte aligned descriptor block. See Section 3.3.2.5 "CommandPtr" on page 22 for details.
3:0	Z	RWU	Undef	**Z:** Indicates the number of contiguous 16-byte aligned blocks at the address pointed to by descriptorAddress. If Z is 0, it indicates that the descriptorAddress is not valid.

Register Descriptions (Continued)

4.4.27 Isochronous Receive

Each isochronous receive context consists of three registers: ContextControl, CommandPtr, and ContextMatch. ContextControl is used by software to control the context's behavior, and is used by hardware to indicate current status. CommandPtr is used by software to tell the IRDMA controller where the DMA context program begins. ContextMatch is used to start on a specified cycle number and to filter received packets based on their tag bits and possible sync bits.

The CS4210 has eight isochronous receive contexts. These registers are repeated at offsets of 20h times the context number. Table 4-52 is a map providing the offset addresses for the ContextControl, CommandPtr, and ContextMatch registers.

4.4.27.1 Isoch Receive Context Control Register

The isochronous receive ContextControl register (see Table 4-53) contains bits that control options, operational state, and status for the isochronous receive DMA contexts. Software can set selected bits by writing ones to the corresponding bits in the ContextControl Set register. Software can clear selected bits by writing ones to the corresponding bits in the ContextControl Clear register. It is not possible for software to set some bits and clear others in an atomic operation. A read from either register returns the same value. It includes several fields which permit software to filter packets based on various combinations of fields within the isochronous packet header.

These registers are repeated at offsets of 20h times the context number (see Table 4-52 for offset address assignment).

Table 4-52. IsochRx Register Address Map

BAR0+Offset	Name
400h	IsochRx0ContextControl Set Register
404h	IsochRx0ContextControl Clear Register
408h	Reserved
40Ch	IsochRx0CommandPtr Register
410h	IsochRx0ContextMatch Register
414h-41Fh	Reserved
420h	IsochRx1ContextControl Set Register
424h	IsochRx1ContextControl Clear Register
428h	Reserved
42Ch	IsochRx1CommandPtr Register
430h	IsochRx1ContextMatch Register
434h-43Fh	Reserved
440h	IsochRx2ContextControl Set Register
444h	IsochRx2ContextControl Clear Register
448h	Reserved
44Ch	IsochRx2CommandPtr Register
450h	IsochRx2ContextMatch Register
454h-45Fh	Reserved
460h	IsochRx3ContextControl Set Register
464h	IsochRx3ContextControl Clear Register
468h	Reserved
46Ch	IsochRx3CommandPtr Register
470h	IsochRx3ContextMatch Register
474h-47Fh	Reserved
480h	IsochRx4ContextControl Set Register
484h	IsochRx4ContextControl Clear Register
488h	Reserved
48Ch	IsochRx4CommandPtr Register
490h	IsochRx4ContextMatch Register
494h-49Fh	Reserved
4A0h	IsochRx5ContextControl Set Register
4A4h	IsochRx5ContextControl Clear Register
4A8h	Reserved
4ACh	IsochRx5CommandPtr Register
4B0h	IsochRx5ContextMatch Register
4B4h-4BFh	Reserved
4C0h	IsochRx6ContextControl Set Register
4C4h	IsochRx6ContextControl Clear Register
4C8h	Reserved
4CCh	IsochRx6CommandPtr Register
4D0h	IsochRx6ContextMatch Register
4D4h-4DFh	Reserved
4E0h	IsochRx7ContextControl Set Register
4E4h	IsochRx7ContextControl Clear Register
4E8h	Reserved
4ECh	IsochRx7CommandPtr Register
4F0h	IsochRx7ContextMatch Register

Register Descriptions (Continued)

Table 4-53. IsochRxnContextControl Set/Clear Register

Bit	Name	Access	Reset	Description
31	bufferFill	RSC	Undef	**Buffer Fill:** When set to one, received packets are placed back-to-back to completely fill each receive buffer (specified by an INPUT_MORE command). When clear, each received packet is placed in a single buffer (described by zero to seven INPUT_MORE commands followed by an INPUT_LAST command). If the multiChanMode bit (bit 28) is set to one, this bit must also be set to one. The value of bufferFill must not be changed while active (bit 10) or run (bit 15) are set to one.
30	isochHeader	RSC	Undef	**Isochronous Header:** When set to one, received isochronous packets include the complete 4-byte isochronous packet header seen by the Link layer. The end of the packet is marked with a xferStatus (bits 15:0 of this register) in the first doublet, and a 16-bit timeStamp indicating the time of the most recently received (or sent) cycleStart packet. When clear, the packet header is stripped off of received isochronous packets. The packet header, if received, immediately precedes the packet payload. Details are in the 1394 OHCI specification data formats. The value of isochHeader must not be changed while active or run are set to one.
29	cycleMatchEnable	RSCU	Undef	**Cycle Match Enable:** In general, when set to one, the context begins running only when the 15-bit cycleMatch field in the corresponding IsochRxnContextMatch register matches the two sets of bits of the bus IsochCycleTimer.cycleSeconds and 13-bit IsochCycleTimer.cycleCount values (BAR0+Offset F0h). The effects of this bit are impacted by the values of other bits in this register and are explained in Section 4.4.26.1 "Isoch Transmit Context Control Register" on page 84. Once the context has become active, hardware clears the cycleMatchEnable bit. The value of cycleMatchEnable must not be changed while active or run are set to one.
28	multiChanMode	RSC	Undef	**Multiple Channel Mode:** When set to one, the corresponding isochronous receive DMA context receives packets for all isochronous channels enabled in the IRChannelMaskHi and IRChannelMaskLo registers (Section 4.4.15 "IRMultiChanMask Registers" on page 69). The isochronous channel number specified in the corresponding IsochRxnContextMatch register is ignored. When set to zero, the IRDMA context receives packets for that single channel. Only one IRDMA context may use the IRChannelMask registers. If more than one IsochRxContextControl register has the multiChanMode bit set, results are undefined. The value of multiChanMode must not be changed while active or run are set to one
27:16	RSVD	--	0	**Reserved**
15	run	RSCU	0	**Run:** The run bit is set by software to enable descriptor processing for a context and cleared by software to stop descriptor processing. The CS4210 only clears this bit on a hardware or software reset. See Section 3.3.2.1 "ContextControl.run" on page 21 for details.
14:13	RSVD	--	0	**Reserved**
12	wake	RSU	Undef	**Wake:** Software sets this bit to 1 to cause the CS4210 to continue or resume descriptor processing. The CS4210 clears this bit on every descriptor fetch. See Section 3.3.2.2 "ContextControl.wake" on page 21 for details.
11	dead	RU	0	**Dead:** The CS4210 sets this bit when it encounters a fatal error and clears this bit when software clears the run bit. See Section 3.3.2.4 "ContextControl.dead" on page 21 for details.
10	active	RU	0	**Active:** The CS4210 sets this bit to 1 when it is processing descriptors. See Section 3.3.2.3 "ContextControl.active" on page 21 for details.
9:8	RSVD	--	0	**Reserved**
7:5	spd	RU	Undef	**Speed:** This field indicates the speed at which the packet was received. 000 = 100 Mbits/sec, 001 = 200 Mbits/sec, and 010 = 400 Mbits/sec. All other values are reserved.
4:0	event code	RW	Undef	**Event Code:** For bufferFill mode, possible values are: ack_complete, evt_descriptor_read, evt_data_write, and evt_unknown. Packets with data errors and packets for which a FIFO overrun occurred are ‘backed-out’ by reverting to the previous state and Xferstatus and resCount are not updated. For packet-per-buffer mode, possible values are: ack_complete, ack_data_error, evt_long_packet, evt_overrun, evt_descriptor_read, evt_data_write, and evt_unknown.

Register Descriptions (Continued)

4.4.27.2 Isoch Receive Command Pointer Register

The CommandPtr register (Table 4-54) specifies the address of the context program which is executed when a DMA context is started. All descriptors are 16-byte aligned, so the four least-significant bits of any descriptor address must be zero. The four least-significant bits of the Command Pointer register are used to encode a Z value that indicates how many physically contiguous descriptors are pointed to by descriptorAddress. In buffer-fill mode, Z is either one or zero. In packet-per-buffer mode, Z is from zero to eight.

These registers are repeated at offsets of 20h times the context number (see Table 4-52 for offset address assignment).

4.4.27.3 Isoch Receive Context Match Register

The ContextMatch register (Table 4-55) is used to start a context running on a specified cycle number, to filter incoming isochronous packets based on tag values and to wait for packets with a specified sync value. All packets are checked for a matching tag value, and a compare on sync is only performed when the descriptor's w field is set to 11. See 1394 OHCI spec data fields for proper usage of the w field. This register should only be written when ContextControl.active is 0, otherwise unspecified behavior results.

These registers are repeated at offsets of 20h times the context number (see Table 4-52 for offset address assignment).

Table 4-54. IsochRxnCommandPtr Register

Bit	Name	Access	Reset	Description
31:4	descriptorAddress	RWU	Undef	**Descriptor Address:** Contains the upper 28 bits of the address of a 16-byte aligned descriptor block. See Section 3.3.2.5 "CommandPtr" on page 22 for details.
3:0	Z	RWU	Undef	**Z:** Indicates the number of contiguous 16-byte aligned blocks at the address pointed to by descriptorAddress. If Z is 0, it indicates that the descriptorAddress is not valid.

Table 4-55. IsochRxnContextMatch Register

Bit	Name	Access	Reset	Description
31	tag3	RW	Undef	**Tag 3:** If set, this context matches on isochronous receive packets with a tag field of 11h.
30	tag2	RW	Undef	**Tag 2:** If set, this context matches on isochronous receive packets with a tag field of 10h.
29	tag1	RW	Undef	**Tag 1:** If set, this context matches on isochronous receive packets with a tag field of 01h.
28	tag0	RW	Undef	**Tag 0:** If set, this context matches on isochronous receive packets with a tag field of 00h.
27	RSVD	--	0	**Reserved**
26:12	cycleMatch	RW	Undef	**Cycle Match:** Contains a 15-bit value, corresponding to the low order two bits of cycleSeconds and the 13-bit cycleCount field in the cycleStart packet. If cycleMatchEnable (bit 29 of corresponding IsochRxnContextControl register) is set, then this IRDMA context becomes enabled for receives when the two low order bits of the bus cycleTime.cycleSeconds and cycleTime.cycleCount (BAR0+Offset F0h) values equal the cycleMatch value.
11:8	sync	RW	Undef	**Synchronous:** This field contains the 4-bit field which is compared to the sync field of each isochronous packet for this channel when the command descriptor's w field is set to 11h.
7	RSVD	--	0	**Reserved**
6	tag1SynFilter	RW	Undef	**Tag 1 Synchronous Filter:** If set and the contextMatch.tag1 bit is set, then packets with tag 01h shall only be accepted into the context if the two most-significant bits of the packet's sync field are 00h. Packets with tag values other than 01h shall be filtered according to the tag0, tag2 and tag3 bits above with no additional restrictions. If clear, this context matches on isochronous receive packets as specified in the tag[0:3] bits above with no additional restrictions.
5:0	channelNum	RW	Undef	**Channel Number:** This 6-bit field indicates the isochronous channel number for which this IRDMA context accepts packets.

Register Descriptions (Continued)

4.5 NATIONAL (NSC) SPECIFIC CONFIGURATION REGISTERS

The NSC configuration registers are at the location specified by the Base Address Register 1 (BAR1) in the PCI configuration space.

The access and operating modes information that is discussed in Section 4.4 "OHCI Configuration Registers" on page 48 apply to these registers.

Table 4-56 is a map for the registers accessed through BAR1. Following this table are subsections providing detailed information for each register.

Table 4-56. NSC Specific Configuration Register Map Summary: BAR1+Offset xxh

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h-03h														**nscControl Register**																	
RSVD										busHoldEnab	PCICapabilities	testOutSel	fastEpromMode	altMode		RSVD	programPhyEnab	aPhyEnhanEnab	linkSpdWp	disableCmDetect	noCycleMaster	altAckConcat	RSVD	altAckAccel	altMultiSpd	altIdleInsert	RSVD	wpDisable	atresbackoff	atreqBackoff	shortCycle
04h-07h														**nscEvent Set Register**																	
RSVD																											eepromAckErr	eepromCRCErr	physLockToggle	disableCmDetect	noCycleMaster
08h-0Bh														**nscEvent Clear Register**																	
RSVD																											eepromAckErr	eepromCRCErr	physLockToggle	disableCmDetect	noCycleMaster
0Ch-0Fh														**nscEventMask Set Register**																	
RSVD																											eepromAckErrMask	eepromCRCErrMask	physLockToggleMask	disableCmDetectMask	noCycleMasterMask
10h-13h														**nscEventMask Clear Register**																	
RSVD																											eepromAckErrMask	eepromCRCErrMask	physLockToggleMask	disableCmDetectMask	noCycleMasterMask
14h-17h														**nscRAMBist**																	
RSVD																							bistFail	bistPass	RSVD						bistGo

Register Descriptions (Continued)

Table 4-56. NSC Specific Configuration Register Map Summary: BAR1+Offset xxh (Continued)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
18h-1Bh	**nscCmcControl**																														
RSVD																														CMCL	CMC
1Ch-1Fh	**Reserved**																														
20h-23h	**nscTxThreshold**																														
isochTxThrsh																asyncTxThrsh															
24h-27h	**nscSubSystem**																														
deviceID																vendorID															
28h-3Fh	**Reserved**																														
40h-43h	**nscPhysReadCount**																														
RSVD																physReadCount															
44h-47h	**nscPhysWriteCount**																														
RSVD																physWriteCount															
48h-4Bh	**nscPhysLockCount**																														
RSVD																physLockCount															
4Ch-5Fh	**Reserved**																														
60h-63h	**nscBusmgrID**																														
busmgrID																															
64h-67h	**nscBandwAvail**																														
bandwAvail																															
68h-6Bh	**nscChanAvailHi**																														
chanAvailHi																															
6Ch-6Fh	**nscChanAvailLo**																														
chanAvailLo																															

Register Descriptions (Continued)

4.5.1 nscControl Register

The nscControl register (Table 4-57) is used to control features of the CS4210 that are not defined within the OpenHCI standard registers. This register may be configured via the serial EEPROM interface.

Table 4-57. BAR1+Offset 00h: nscControl Register

Bit	Name	Access	Reset	Description
31:22	RSVD	---	0	**Reserved**
21	busHoldEnab	RW	0	**Bus Hold Enable:** Allow bus holders for single capacitor isolation. 0 = Disable; 1 = Enable.
20	PCICapabilities	RW	0	**PCI Capabilities:** Allow the PCI power management structures to be visible in the PCI configuration header. 0 = Disable; 1 = Enable. See Table 4-6: "Index 44h Capability ID Register" on page 45, "Index 46h-47h Power Management Capabilities Register" on page 46, and "Index 48h Power Management and Control Status Register" on page 46.
19	testOutSel	RW	0	**Test Out Pin Select:** Selects function of TESTO pin (pin 7). 0 = TESTO is the output of the NAND tree. 1 = TESTO pin reflects RAM BIST status.
18	fastEpromMode	RW	0	**Fast EEPROM Mode:** Allow fast operation of the EEPROM interface for test purposes. 0 = Disable; 1 = Enable.
17:16	altMode	RW	00	**Alternate Mode:** Modifier for aPhyEnhanceEnable and the alt bits in this register. See Table 4-58 "altMode" on page 92 for details.
14	programPhyEnab	RW	0	**Program PHY Enable:** Mirror of HCControl.programPhyEnable bit (BAR0+Offset 50h[23]).
13	aPhyEnhanEnab	RW	0	**A PHY Enhancement Enable:** Mirror of HCControl.aPhyEnhanceEnable bit (BAR0+Offset 50h[22]).
12	linkSpdWp	RW	0	**Link Speed Write Protect:** Makes BusOptions.link_spd field (BAR0+Offset 20h[2:0]) read only. 0 = Disable; 1 = Enable.
11	disableCmDetect	RW	0	**Disable Cycle Master Detector:** Turn-off the no cycle master detector logic. 0 = Disable; 1 = Enable. When this bit is set, the link layer logic that detects the absence of a cycle master is disabled. This detector is not a standard feature of the OpenHCI specification, thus this bit is defined to handle any unforeseen interoperability issues. If the noCycleMaster bit is also set, then setting this bit will have no effect.
10	noCycleMaster	RWU	0	**No Cycle Master:** Force an internal “no cycle master present” condition. 0 = Disable; 1 = Enable. Setting this bit overrides the cycle master detector logic. When this bit is clear, the link layer must observe 255 consecutive cycle lost events before concluding that no cycle master node is present.
9	altAckConcat	RW	0	**Alternate Acknowledge Concatenation:** Modify the interpretation of the HCControl.aPhyEnhanceEnable bit (BAR0+Offset 50h[23]) with respect to concatenation of packets onto the transmission of handshake packets. See altMode (bits [17:16]) and Table 4-58 "altMode" on page 92 for details.
8	RSVD	--	0	**Reserved**
7	altAckAccel	RW	0	**Alternate Acknowledge Accelerate:** Modify the interpretation of the HCControl.aPhyEnhanceEnable bit (BAR0+Offset 50h[23]) with respect to ACK accelerated arbitration. When enabled the link layer will not consider the media “lost” when receiving a handshake packet. See altMode (bits [17:16]) and Table 4-58 "altMode" on page 92 for details.
6	altMultiSpd	RW	0	**Alternate Multispeed:** Modify the interpretation of the HCControl.aPhyEnhanceEnable bit (BAR0+Offset 50h[23]) with respect to multispeed packet concatenation. See altMode (bits [17:16]) and Table 4-58 "altMode" on page 92 for details.
5	altIdleInsert	RW	0	**Alternate Idle Insert:** Modify the interpretation of the HCControl.aPhyEnhanceEnable bit (BAR0+Offset 50h[23]) with respect to the signaling performed by the link when it has been granted permission to transmit. See altMode (bits [17:16]) and Table 4-58 "altMode" on page 92 for details.
4	RSVD	--	0	**Reserved**

Register Descriptions (Continued)

Table 4-57. BAR1+Offset 00h: nscControl Register (Continued)

Bit	Name	Access	Reset	Description
3	wpDisable	RW	0	**Write Protect Disable:** Turn-off write protection of selected vendor specific registers. 0 = Disable; 1 = Enable.
2	atresBackoff	RW	0	**AT Response Context Backoff:** Allow backoff timing for retransmission of asynchronous requests via the AT response context. 0 = Disable; 1 = Enable.
1	atreqBackoff	RW	0	**AT Request Context Backoff:** Allow backoff timing for retransmission of asynchronous requests via the AT request context. 0 = Disable; 1 = Enable.
0	shortCycle	RW	0	**Short Cycle:** Allow cycle count to increment when cycle offset = 511 instead of 3071. 0 = Disable; 1 = Enable. Short cycle operation is intended for testing only.

The altMode bits provide fine control over the individual P1394a features. The individual bit controls for altIdleInsert, altMultiSpd, altAckAccel, and altAckConcat can be modified by controlling the altMode bits. These four individual bits are considered the modifier in the Table 4-57. The combination of the individual alt bits and the altMode determine the enabling of the P1394a feature. For example, the second line in the table represents the case where P1394a features are enables with no modification. The result field being one indicates that the P1394a feature is enabled. For the case of the fourth row, if the altAckAccel bit were set to one for example, then that single P1394a feature would be disabled as indicated by the zero in the result column.

Table 4-58. altMode

nscControl.altMode (BAR1+Offset 00h[17:16]	HCControl.aPhyEnhance Enable (BAR0+Offset 50h[22])	nscControl.Modifier (BAR1+Offset 00h[9, 7, 6, 5])	Result
00 or 11 (XOR)	0	0	0
00 or 11	1	0	1
00 or 11	0	1	1
00 or 11	1	1	0
01 (AND)	0	0	0
01	1	0	0
01	0	1	0
01	1	1	1
10 (OR)	0	0	0
10	1	0	1
10	0	1	1
10	1	1	1

Register Descriptions (Continued)

4.5.2 nscEventSet/Clear

The nscEvent register (Table 4-59) contains events that are unique to the CS4210. This register is set (BAR1+Offset 04h) and clear (BAR1+Offset 08h).

4.5.3 nscMaskSet/Clear

The nscEventMask register (Table 4-60) controls the masking of the nscEvent register. This register is set (BAR1+Offset 0Ch) and clear (BAR1+Offset 10h).

Table 4-59. BAR1+Offset 04h (Set) and 08h (Clear): nscEvent Register

Bit	Name	Access	Reset	Description
31:5	RSVD	--	0	**Reserved**
4	eepromAckErr	RSCU	Undef	**EEPROM Acknowledge Error:** Set when an acknowledge is missing from the serial EEPROM load.
3	eepromCRCErr	RSCU	Undef	**EEPROM Cyclical Redundancy Check Error:** Set when a CRC error occurs as a result of the serial EEPROM load.
2	physLockToggle	RSCU	Undef	**Physical Lock Toggle:** Set when D15 of the nscPhysLockCount register (BAR1+Offset 48h) changes.
1	disableCmDetect	RSCU	Undef	**Disable Cycle Master Detector:** Set when D15 of the nscPhysWriteCount register (BAR1+Offset 44h) changes.
0	noCycleMaster	RSCU	Undef	**No Cycle Master:** Set when D15 of the nscPhysReadCount register (BAR1+Offset 40h) changes.

Table 4-60. BAR1+Offset 0Ch (Set) and 10h (Clear): nscEventMask Register

Bit	Name	Access	Reset	Description
31:5	RSVD	--	0	**Reserved**
4	eepromAckErrMask	RSCU	Undef	**EEPROM Acknowledge Error Mask:** Set to 1 enables the corresponding bit in the nscEvent register (BAR1+Offset 04h).
3	eepromCRCErrMask	RSC	Undef	**EEPROM Cyclical Redundancy Check Error Mask:** Set to 1 enables the corresponding bit in the nscEvent register (BAR1+Offset 04h).
2	physLockToggleMask	RSC	Undef	**Physical Lock Toggle Mask:** Set to 1 enables the corresponding bit in the nscEvent register (BAR1+Offset 04h).
1	disableCmDetectMask	RSC	Undef	**Disable Cycle Master Detector Mask:** Set to 1 enables the corresponding bit in the nscEvent register (BAR1+Offset 04h).
0	noCycleMasterMask	RSC	Undef	**No Cycle Master Mask:** Set to 1 enables the corresponding bit in the nscEvent register (BAR1+Offset 04h).

Register Descriptions (Continued)

4.5.4 nscRAMBist

The nscRAMBist register (Table 4-61) is used to test the FIFOs implemented in SRAM. The bistGo bit is set and the bistFail and bistPass bits are polled until one is set, indicating pass or fail. The CS4210 must be reset before normal operation can resume.

4.5.5 nscCmcControl

The nscCmcControl register (Table 4-62) controls the state of the CMC and CMCL pins. These pins are intended for use with Contender pins on 1394-1995 PHY devices. This feature is not used on the CS4103.

4.5.6 nscTxThreshold

The nscTxThreshold register (Table 4-63) is used to control the point at which the transmit FIFOs start sending data over the 1394 bus. Lowering the threshold value causes smaller packets to be sent. This register can only be written to when the nscControl.wpDisable bit (BAR1+Offset 00h[3]) is set. This register may be configured via the serial EEPROM interface (if EEPROM is present).

4.5.7 nscSubSystem

The nscSubSystem register (Table 4-64) is used to configure the subsystem vendor ID and the subsystem device ID values in the PCI configuration space. This register can only be written to when the nscControl.wpDisable bit (BAR1+Offset 00h[3]) is set. This register may be configured via the serial EEPROM interface (if EEPROM is present).

Table 4-61. BAR1+Offset 14h: nscRAMBist Register

Bit	Name	Access	Reset	Description
31:10	RSVD	--	0	**Reserved**
9	bistFail	RU	Undef	**Built-in Self-Test Fail:** Set to 1 when RAM BIST has failed.
8	bistPass	RU	Undef	**Built-in Self-Test Pass**: Set to 1 when RAM BIST has passed.
7:1	RSVD	--	0	**Reserved**
0	bistGo	RW	Undef	**Built-in Self-Test Go:** RAM BIST is started by writing a 1 to this bit.

Table 4-62. BAR1+Offset 18h: nscCmcControl Register

Bit	Name	Access	Reset	Description
31:2	RSVD	--	0	**Reserved**
1	CMCL	RW	0	**Contender Master Control Link Enabled:** This bit is ANDed with HCControl.LinkEnable (BAR0+Offset 50h[17]) and the result is reflected on pin 3.
0	CMC	RW	0	**Contender Master Control:** Selects polarity of CMC output. This bit is directly reflected on pin 2. 0 = Low; 1 = High.

Table 4-63. BAR1+Offset 20h: nscTxThreshold Register

Bit	Name	Access	Reset	Description
31:16	isochTxThrsh	RW	01FEh	**Isochronous Transmit Threshold:** FIFO threshold value when transmitting isochronous packets. D[18:16] are hardwired to 110 and D[31:25] are hardwired to 00h.
15:0	asyncTxThrsh	RU	01FEh	**Asynchronous Transmit Threshold:** FIFO threshold value when transmitting asynchronous packets. D[2:0] are hardwired to 110 and the D[15:9] are hardwired to 00h.

.

Table 4-64. BAR1+Offset 24h: nscSubSystem Register

Bit	Name	Access	Reset	Description
31:16	deviceID	RW	0000h	**Device ID:** PCI configuration subsystem device ID.
15:0	vendorID	RW	0000h	**Vendor ID:** PCI configuration subsystem vendor ID.

Register Descriptions (Continued)

4.5.8 nscPhysReadCount

The nscPhysReadCount register (Table 4-65) provides statistics on physical read requests and CSR read requests. This register is cleared when a 1 is written to D0.

4.5.9 nscPhysWriteCount

The nscPhysWriteCount register (Table 4-66) provides statistics on physical write requests. This register is cleared when a 1 is written to D0.

4.5.10 nscPhysLockCount

The nscPhysLockCount register (Table 4-67) provides statistics on physical lock requests. This register is cleared when a 1 is written to D0.

Table 4-65. BAR1+Offset 40h: nscPhysReadCount

Bit	Name	Access	Reset	Description
31:16	RSVD	--	0	**Reserved**
15:0	physReadCount	RW	0000h	**Physical Read Count:** Current count of physical read requests and/or CSR read requests.

Table 4-66. BAR1+Offset 44h: nscPhysWriteCount Register

Bit	Name	Access	Reset	Description
31:16	RSVD	--	0	**Reserved**
15:0	physWriteCount	RW	0000h	**Physical Write Count:** Current count of physical write requests.

Table 4-67. BAR1+Offset 48h: nscPhysLockCount Register

Bit	Name	Access	Reset	Description
31:16	RSVD	--	0	**Reserved**
15:0	physLockCount	RW	0000h	**Physical Lock Count:** Current count of physical lock requests.

Register Descriptions (Continued)

4.5.11 nscBusmgrID

The nscBusmgrID register (Table 4-68) is a directly readable BUS_MANAGER_ID CSR resource.

4.5.12 nscBandwAvail

The nscBandwAvail register (Table 4-69) is a directly readable BANDWIDTH_AVAILABLE CSR resource.

4.5.13 nscChanAvailHi

The nscChanAvailHi register (Table 4-70) is a directly readable CHANNELS_AVAILALBLE_HI CSR resource.

4.5.14 nscChanAvailLo

The nscChanAvailLo register (Table 4-71) is a directly readable CHANNELS_AVAILABLE_LO CSR resource.

Table 4-68. BAR1+Offset 60h: nscBusmgrID Register

Bit	Name	Access	Reset	Description
31:0	busmgrID	R	Undef	**Bus Manager ID:** BUS_MANAGER_ID CSR resource.

Table 4-69. BAR1+Offset 64h: nscBandwAvail Register

Bit	Name	Access	Reset	Description
31:0	bandwAvail	R	Undef	**Bandwidth Available:** BANDWIDTH_AVAILABLE CSR resource.

Table 4-70. BAR1+Offset 68h: nscChanAvailHi Register

Bit	Name	Access	Reset	Description
31:0	chanAvailHi	R	Undef	**Channels Available High:** CHANNELS_AVAILALBLE_HI CSR resource.

Table 4-71. BAR1+Offset 6Ch: nscChanAvailLo Register

Bits	Field Name	Access	Reset	Description
31:0	chanAvailLo	R	Undef	**Channels Available Low:** CHANNELS_AVAILALBLE_LO CSR resource.

5.0 Electrical Specifications

This section provides information on NAND tree test mode, absolute maximum ratings, recommended operating conditions, and DC/AC characteristics for the Geode CS4210.

5.1 NAND TREE TEST MODE

The NAND tree test mode is used to test input and bidirectional pins which are part of the NAND tree chain. After power is applied to the CS4210, pin 7 (TESTO) is the output of the NAND tree. To test the NAND tree, all inputs in Table 5-1 are held low and then a high is successively applied to the inputs in the order listed in Table 5-1. The TESTO output will toggle on each input change as shown in Figure 5-1.

The TESTO pin will continue to output the result of the NAND tree during normal operation. If this is undesirable, nscControl.testOutSel (BAR1+Offset 00h[19]) can be set and TESTO will reflect the BIST output which does not toggle during normal operation. The nscControl register is loaded via the serial EEPROM upon power up, so nscControl.testOutSel can be set as the default if desired.

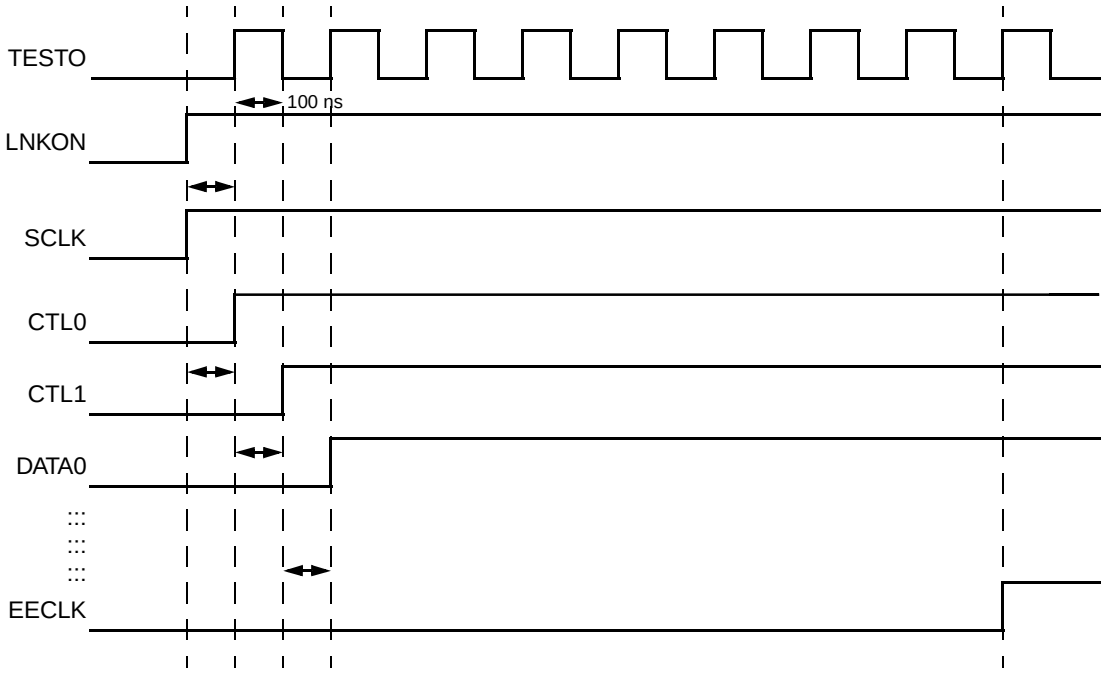
Table 5-1. NAND Tree Test Mode Pins

Signal Name	Pin No.
LNKON	98
SCLK	95
CTL0	93
CTL1	92
DATA0	90
DATA1	89
DATA2	88
DATA3	86
DATA4	85
DATA5	84
DATA6	82
DATA7	81
DIRECT	79
CCLKI	78
TESTEN#	76
AD0	74
AD1	73

Signal Name	Pin No.
AD2	72
AD3	71
AD4	69
AD5	68
AD6	67
AD7	66
C/BE0#	65
AD8	64
AD9	62
AD10	61
AD11	59
AD12	58
AD13	57
AD14	56
AD15	54
C/BE1#	53
PAR	52

Signal Name	Pin No.
SERR#	51
PERR#	49
STOP#	48
DEVSEL#	47
TRDY#	45
IRDY#	44
FRAME#	43
C/BE2#	41
AD16	40
AD17	38
AD18	37
AD19	36
AD20	34
AD21	33
AD22	32
AD23	31
IDSEL	29

Signal Name	Pin No.
C/BE3#	28
AD24	27
AD25	26
AD26	25
AD27	23
AD28	22
AD29	21
AD30	19
AD31	18
PME#	17
PGNT#	14
PCLK	12
RST#	10
EEDATA	5
EECLK	4



Note: The following pins are not in the NAND tree: CMC, CMCL, INTA#, PREQ#, CCLKO, LREQ, LPS, and all supplies.

Figure 5-1. NAND Tree Output Waveform

Electrical Specifications (Continued)

5.2 ABSOLUTE MAXIMUM RATINGS

Table 5-2 lists the absolute maximum ratings for the CS4210. Stresses beyond the listed ratings may cause permanent damage to the device. Exposure to conditions beyond these limits may (1) reduce device reliability and (2) result in premature failure even when there is no immediately apparent sign of failure. Prolonged exposure to conditions at or near the absolute maximum ratings may also reduce useful life and reliability. These are stress ratings only and do not imply that operation under any conditions other than those listed in Table 5-3 is possible.

5.3 OPERATING CONDITIONS

All DC and AC parameters were measured under the operating conditions listed in Table 5-3.

Table 5-2. Absolute Maximum Ratings

Parameter	Units
Supply Voltage, V_{DD}, V_{DDIO}	–0.5V to 3.75V
Input Voltage	–0.5V to V_{DDIO} + 1.7V
Output Voltage	–0.5V to V_{DDIO} + 0.5V
Storage Temperature, T_{STG}	–65°C to 150°C
ESD Tolerance	2000V
Lead Temperature, T_L	230°C (Soldering 10 seconds)

Table 5-3. Operating Conditions

Parameter	Units
I/O Supply Voltage, V_{DDIO}	3.0V to 3.6V
Supply Voltage, V_{DD}	2.3V to 2.7V
Operating Temperature, T_A	0°C to 70°C
Power Dissipation, P_D	300 mW

Electrical Specifications (Continued)

5.4 DC CHARACTERISTICS

Table 5-4. DC Characteristics: PCI Interface

Symbol	Parameter	Min	Max	Units	Conditions
V_{DDIO}	I/O Supply Voltage	3.0	3.6	V	
V_{DD}	Core Supply Voltage	2.3	2.7	V	
V_{IH}	Input High Voltage	1.7		V	
V_{IL}	Input Low Voltage	–0.5	0.7	V	
V_{IPU}	Input Pull-up Voltage	1.7		V	
I_{IH}	Input High Leakage Current		+ 10	µA	$0 < V_{IN} < V_{DDIO}$
I_{IL}	Input Low Leakage Current		–10	µA	$0 < V_{IN} < V_{DDIO}$
V_{OH1}	Output High Voltage	V_{DDIO}–0.2		V	I_{OUT} = –10 µA
V_{OH2}	Output High Voltage	2.4		V	I_{OUT} = –2 mA
V_{OL1}	Output Low Voltage		0.2	V	I_{OUT} = 10 µA
V_{OL2}	Output Low Voltage		0.4	V	I_{OUT} = 2 mA
C_{IN}	Input Pin Capacitance		10	pF	
C_{CLK}	CLK Pin Capacitance	5	12	pF	
C_{IDSEL}	IDSEL Pin Capacitance		8	pF	
L_{PIN}	Pin Inductance		20	nH	

Table 5-5. DC Characteristics: General

Symbol	Parameter	Min	Typ	Max	Units	Conditions
V_{DDIO}	I/O Supply Voltage	3.0		3.6	V	
V_{DD}	Core Supply Voltage	2.3		2.7	V	
I_{DDQ}	Quiescent Core Supply Current		100		µA	
I_{DDQIO}	Quiescent I/O Supply Current		500		µA	
I_{DD}	Dynamic Core Supply Current		80		mA	
I_{DDIO}	Dynamic Supply Current		13		mA	

Electrical Specifications (Continued)

Table 5-6. IEEE 1394a PHY-Link Interface

Symbol	Parameter	Min	Max	Units	Conditions
V_{DDIO}	Supply Voltage	3.0	3.6	V	
V_{DD}	Core Supply Voltage	2.3	2.7	V	
V_{IH}	Input High Voltage	2.8	V_{DD}+10%	V	
V_{IL}	Input Low Voltage	GND	0.7	V	
I_{IH}	Input High Leakage Current		40	µA	
I_{IL}	Input Low Leakage Current		600	µA	
V_{OH}	Output High Voltage	2.8	V_{DD}	V	I_{OUT} = –4 mA
V_{OL}	Output Low Voltage	GND	0.4	V	I_{OUT} = 4 mA
C_{IN}	Input Pin Capacitance		7.5	pF	

Electrical Specifications (Continued)

5.5 AC SPECIFICATIONS

Table 5-7. PCI Timings: C_{LOAD} = 50 pF

Symbol	Parameter	Min	Max	Units
t_{VAL}	CLK to Signal Valid Delay - bused signals	2	11	ns
t_{ON}	Float to Active Delay	2		ns
t_{OFF}	Active to Float Delay		28	ns
t_{SU}	Input Setup Time to CLK - bused signals	7		ns
t_{H}	Input Hold Time from CLK	0		ns
t_{RST}	Reset Active Time after Power Stable	1		ms
$t_{RST-CLK}$	Reset Active Time after CLK Stable	100		ms
$t_{RST-OFF}$	Reset Active Time to Output TRI-STATE Delay		40	ns
t_{CYC}	PCLK Cycle Time	30		ns
t_{HIGH}	PCLK High Time	11		ns
t_{LOW}	PCLK Low Time	11		ns

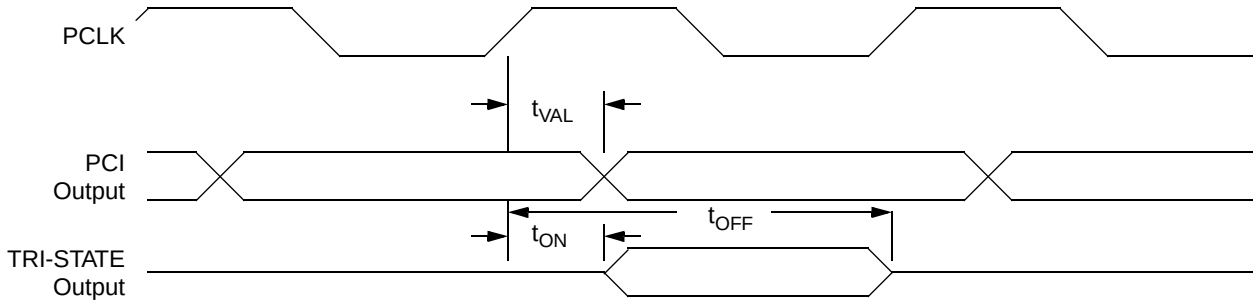


Figure 5-2. PCI Timing Waveform

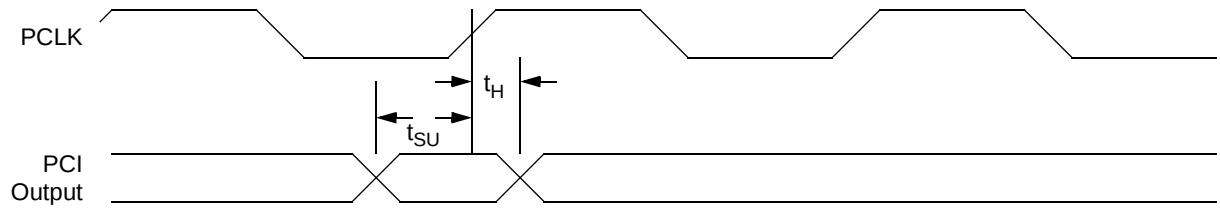


Figure 5-3. PCI Setup and Hold Timing

6.0 Physical Dimensions

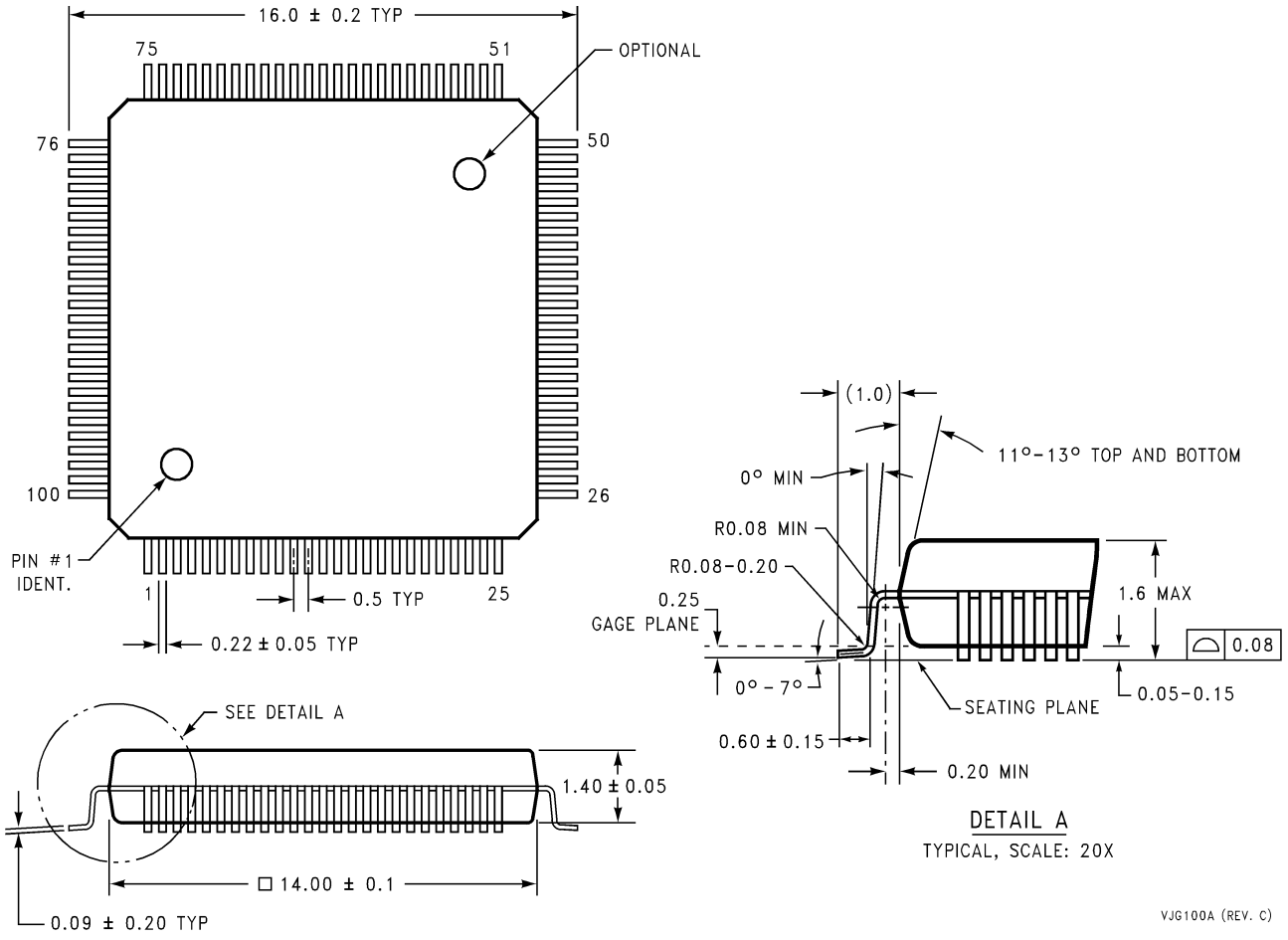


Figure 6-1. 100-Pin LQFP (Low-Profile Quad Flat Pack) Package
Order Number: CS4210VJG

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com

National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 87 90

National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507
Email: nsj.crc@jksmtp.nsc.com