SharcFIN ASIC

SFIN-160 For ADSP-21160 SHARC DSPs

User's Manual

BittWare

SharcFIN ASIC

User's Manual for the SFIN-160 SharcFIN

December 3, 2003 Edition

31 B South Main St.
Concord, NH 03301
Phone: 603.226.0404
Fax: 603.226.6667
Web: www.bittware.com

Contents

Chapter 1
Introduction

Chapter 2
Functional Description

Chapter 3
Register Descriptions

Chapter 4
Hardware Description

Appendix A
Timing Diagrams

List of Figures

List of Tables

Chapter 1
Introduction

The SFIN-160 SharcFIN ASIC is a feature-rich, single device that combines the complex control of the PCI bus with an interface to the Analog Devices' ADSP-21160 SHARC® DSP. The SharcFIN ASIC flexibly interfaces the ADSP-21160 DSPs to a wide range of interfaces, including 64/66 MHz PCI bus (rev. 2.2 compliant), SDRAM, UART, I^2C™ serial ports, flash memory, and a general-purpose expansion bus. The SharcFIN also provides a feature-rich set of DMA functions and interrupt options to support very high-speed, real-time data flow with a minimum of processor overhead.

This chapter discusses the following topics:

1.1 Functional Overview of the SharcFIN

This section provides an overview of how the SharcFIN functions, discussing its features, architecture, and data flow.

1.1.1 SharcFIN Features

The following is a list of the SharcFIN's features:

- 64-bit, 66 MHz PCI rev. 2.2 compliant interface (528 MB/s burst)
- Interface to 64-bit, 40 MHz ADSP-21160 cluster bus
- Peripheral bus interface
  - 8 bits wide @ 20 MHz
  - Accessible from the ADSP-21160 cluster bus and the PCI bus
  - Flash interface for ADSP-21160 boot and non-volatile data storage
- Six independent FIFOs (2.4 KB total)
  - Four DMA buffers, 64×64 each (two transmit, two receive)
  - Two target buffers, 32×64 write, 16×64 read
- Direct, single PCI access from the ADSP-21160 cluster bus
- 16-byte configurable PCI mailbox registers
- I_2O™ V1.5 compliant
- Programmable interrupt multiplexer: 10 inputs, 7 outputs (one of each dedicated to PCI)
- SDRAM controller on ADSP-21160 cluster bus; supports up to 512 MB
- Standard UART and I^2C interface

1.1.2 SharcFIN Architecture Overview

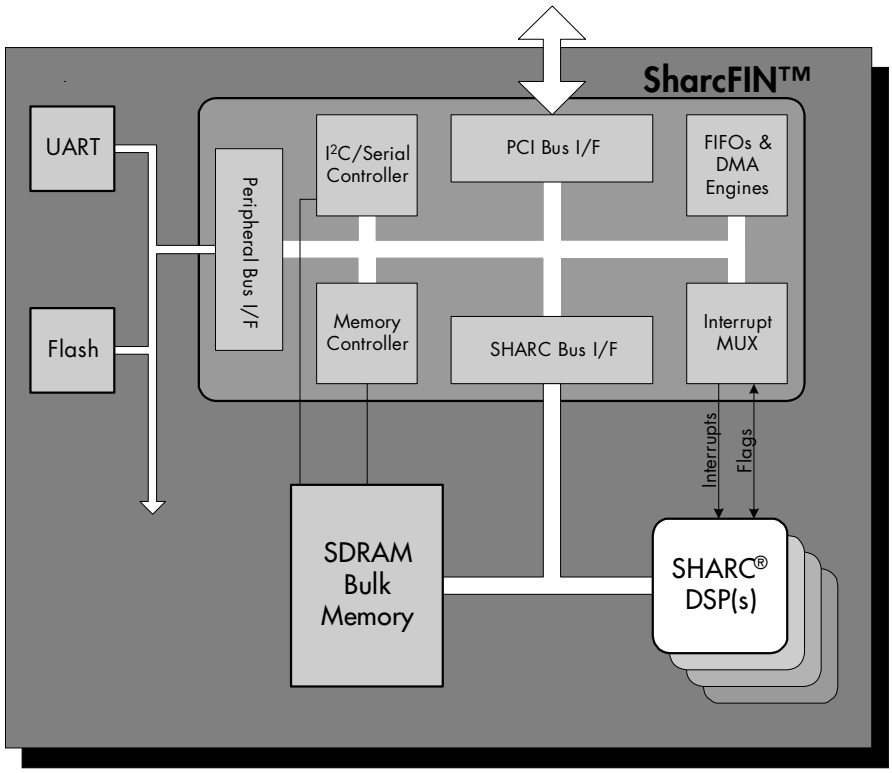
This section provides an overview of the SharcFIN architecture. Figure 1-1 on page 1–3 is a simplified block diagram of the SharcFIN architecture as it is implemented on an ADSP-21160 SHARC DSP board.

The SharcFIN ASIC flexibly interfaces the ADSP-21160 DSPs to a wide range of interfaces, including 64/66 MHz PCI bus (rev. 2.2 compliant), SDRAM, UART, I^2C serial ports, flash memory, and a general-purpose expansion bus (the 8-bit peripheral bus). The SharcFIN also provides a feature-rich set of DMA functions and interrupt options to support very high-speed, real-time data flow with a minimum of processor overhead.

The SharcFIN consists of two main sections: the PCI interface and the SHARC interface. The *PCI interface* consists of a full 64-bit, 66 MHz bus mastering PCI interface. It includes two DMA transmit channels, two DMA receive channels, and a single PCI read/write channel. Also included in the PCI interface is full I^20 support with the associated mailboxes.

The *SHARC interface* of the SharcFIN provides the ADSP-21160 specific functionality, which includes the SDRAM interface and control, the Flash and dual UART, the interrupt multiplexers, and the I^2C interface.

Figure 1-1
Simplified Block Diagram of the SharcFIN ASIC



Interface to ADSP-21160 Cluster Bus

The first function of the SharcFIN is to interface to the ADSP-21160 cluster bus. The SharcFIN provides a 64-bit interface to the ADSP-21160 cluster bus. It also integrates a full-featured SDRAM controller, which allows the ADSP-21160s to access SDRAM using burst mode access at sustained data rates of 256 Mbytes/sec for writes and 180 Mbytes/sec for reads.

Interface to PCI

The second function of the SharcFIN is to interface to PCI. The SharcFIN implements a full 64-bit/66MHz master PCI interface. The PCI interface is PCI rev 2.2 compliant and provides 16 Bytes of configurable PCI mailbox registers.

Interface to Peripheral Bus

A third bus interface is provided by the SharcFIN's peripheral bus. The peripheral bus is a general-purpose utility bus that allows easy interface to standard microprocessor peripherals such as UARTs and flash memory. It provides a simple, glueless way to add additional functionality to your SHARC DSP board.

SDRAM Controller

The SharcFIN also implements a full-featured SDRAM controller, which supports up to 512 Mbytes of SDRAM. It refreshes the SDRAM and controls all of the interfacing from the ADSP-21160s to the SDRAM and from the PCI interface to the SDRAM.

I^2C Interface

The SharcFIN's I^2C interface allows it to communicate with a wide variety of integrated circuits. Uses include data communications, SharcFIN interconnection, and hardware configuration and identification. BittWare Hammerhead DSP boards use the I^2C interface with serial EEPROMs containing configuration information and some user non-volatile storage space.

Interrupt Multiplexer

The SharcFIN integrates an extensive interrupt and flag multiplexer to facilitate system-level control and coordination of multiprocessors. This programmable resource allows each ADSP-21160 to select the sources of its hardware interrupts; sources include other processors, PCI, peripherals, and the internal DMA engines.

1.1.3 SharcFIN Data Flow Overview

Figure 1-2 on page 1–7 illustrates the data flow through the SharcFIN. The SharcFIN's PCI interface logic provides a 64-bit, 66 MHz PCI interface. The SharcFIN has six independent FIFOs. All data moving between the PCI and ADSP-21160 cluster busses passes through one of these FIFOs. The FIFOs are independent: pointers and memory are not shared, dramatically increasing the amount of possible bandwidth through the SharcFIN.

PCI-to-DSP Target Transactions

Target transactions in which the PCI host is the master and the ADSP-21160 is the target use the Target Read/Write Prefetch FIFOs. The Target Read/Write Prefetch FIFOs are buffers for target transactions from the PCI bus to the ADSP-21160 cluster bus. (See section 2.1 for a description of how to use the SharcFIN's PCI-to-DSP target interface.)

Target Write/Post FIFO

The Target Write/Post FIFO is 72 bits wide by 32 deep. It is used by the target controller to buffer target writes to the SharcFIN. The FIFO holds 64 bits of data and 8 bits of byte lane information.

Note *The user does not have direct access to the Target Write Post FIFO, since the target controller automatically extracts the address and data.*

Target Read/Prefetch FIFO

The Target Read/Prefetch FIFO is 64 bits wide by 16 deep. It is used by the target controller to hold data for the target reads and target prefetches, and it holds 64 bits of data.

DSP-to-PCI Target Transactions

Single target transactions in which the ADSP-21160 is the master and the PCI is the target use single PCI access. Single PCI Access allows the ADSP-21160 to master single word transfers on the PCI bus. A local processor may transfer single quadwords of data to/from PCI memory by reading and writing to the Single PCI (SPCI) registers in the control space. This mode is also used to do Type 0/1 Configuration Cycles. FIFOs are not used. (See section 2.2 for a description of how to use the DSP-to-SharcFIN target interface, and see "Single PCI Accesses" on page 43 for a description of Single PCI Accesses.)

DMA Transactions

DMA transactions use Transmit FIFO 0/1 and Receive FIFO 0/1. Transmit FIFO 0/1 and Receive FIFO 0/1 are buffers to hold data from the DMA channels as it transfers from bus to bus.

Transmit FIFO0

Transmit FIFO0 is 72 bits wide by 64 deep. It is used to move data from the local busses to PCI under the control of Transmit Channel Master Controller 0. This FIFO holds 64 bits of data and 8 bits of byte lane information.

Transmit FIFO1

Transmit FIFO1 is 72 bits wide by 64 deep. It is used to move data from the local busses to PCI under the control of Transmit Channel Master Controller 1. This FIFO holds 64 bits of data and 8 bits of byte lane information.

Receive FIFO0

Receive FIFO0 is 72 bits wide by 64 deep. It is used to move data from the PCI to the local busses under the control of Receive Channel Master Controller 0. This FIFO holds 64 bits of data, 8 bits of byte lane information.

Receive FIFO1

Receive FIFO1 is 72 bits wide by 64 deep. It is used to move data from PCI to the local busses under the control of Receive Channel Master Controller 1. This FIFO holds 64 bits of data and 8 bits of byte lane information.

The two transmit FIFOs and the two receive FIFOs have several different modes of operation. The following is a brief description of the different modes.

Direct FIFO Mastering

A PCI starting address and transfer count are set by the host or local processor, and data is automatically transferred by the FIFOs to/from PCI memory. Because each FIFO has its own controller, two transmit and two receive channels can be running simultaneously.

Chain Mode DMA

Transmit FIFO0 and Receive FIFO0 can be dedicated to executing a batch of transmit/receive mastering operations. These operations are also called "*scatter gather DMA*" and "*shuttle mode DMA*."

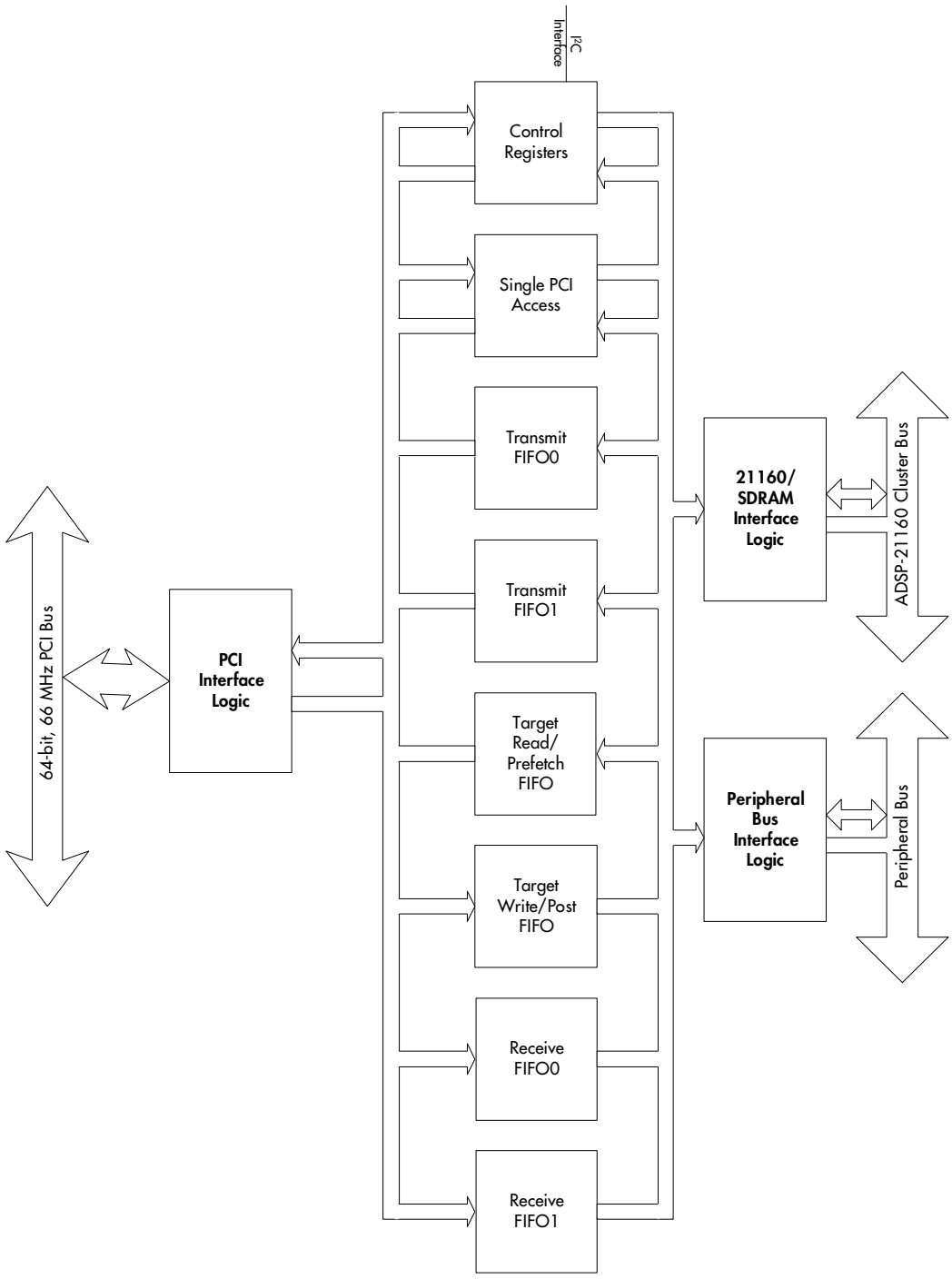
Control Registers

A set of control registers controls most of the function of the SharcFIN chip, including PCI control registers, I^2C interfaces, DMAs, mode registers, and the interrupt multiplexers. The PCI bus and the ADSP-21160 bus both have read and write access to these registers.

SDRAM Controller and Peripheral Bus Interface

The ADSP-21160 SDRAM interface logic provides an interface to SDRAM for both the PCI host and the ADSP-21160 DSP. The peripheral bus interface logic provides an interface to flash memory, a dual UART, and any other peripheral bus devices.

Figure 1-2
SharcFIN Data Flow Overview



1.2 About this Document

This section provides general information about the *SharcFIN ASIC User's Manual*. It discusses the following topics:

- The contents and purpose of this document (see section 1.2.1)
- The conventions used throughout this document (see section 1.2.2)
- The revision history of this document (see section 1.2.3)
- Other documents related to this user's manual (see section 1.2.4)

1.2.1 Contents and Purpose of This Document

What This Document Covers

This document is a complete user's guide and reference for BittWare's SFIN-160 SharcFIN ASIC for the ADSP-21160 SHARC DSP. It provides the information you'll need to understand how the SharcFIN functions and how to program it. It contains the following information:

- Descriptions of the features and architecture of the SharcFIN (Chapter 1)
- Detailed description of how the SharcFIN functions (Chapter 2)
- Register descriptions and memory maps for the SharcFIN (Chapter 3)
- Pinouts and signal descriptions for the SharcFIN (Chapter 4)
- Electrical specifications and timing diagrams for the SharcFIN (Appendix A)

What You Are Expected To Know

This document assumes that you have prior knowledge of the following:

- The architecture and function of the Analog Devices ADSP-21160 SHARC DSP
- C and assembly language development on the ADSP-21160 SHARC DSP
- The *PCI Local Bus Specification*, Revision 2.2

1.2.2 Conventions Used in This Document

This section explains the document conventions we have used throughout the *SharcFIN ASIC User's Manual*.

Addressing Conventions

Because the ADSP-21160 SHARC DSPs use 32-bit word addressing, unless otherwise noted, all addresses in this manual are 32-bit word addresses.

Signal Notation

Signals are either active high or active low. Active low signals are defined as true (asserted) when they are at a logic low. Similarly, active high signals are defined as true at a logic high.

- Each signal that assumes a logic low state when asserted is designated with an "overbar." For example, $\overline{\text{SIGNAL}}$ is asserted low to indicate an active signal.
- Signals that are not designated with an overbar are asserted when they assume the logic high state. For example, SIGNAL is asserted high to indicate an active signal.

Object Size Notation

The following designations are used throughout this manual when referring to the size of data objects:

- A *byte* is an 8-bit object.
- A *word* is a 16-bit, or 2-byte object.
- An *lword* or a *dword* is a long word and is a 32-bit or 4-byte object.
- A *dlword* or *quadword* is a double long word and is a 64-bit or 8-byte object.

Bit Ordering Conventions

This document adopts the convention that the most significant bit is always the largest number (also referred to as *Little-Endian* bit ordering). For example, in a register that consists of bit_name[31:0], bit_name[31] is the most significant bit, and bit_name[0] is the least significant bit of the register.

Numeric Conventions

We have used the following numeric conventions:

- Hexadecimal numbers are designated by the prefix *0x*. For example, 0x01A0.
- Binary numbers are designated by the prefix *0b*. For example, 0b1010.
- Decimal numbers have no prefix. For example, 4356.

Abbreviations

In the signal description tables (Table 4-1 on page 4-196 and Table 4-2 on page 4-200), for the interface between the SHARC interface and the PCI core, the following abbreviations are used:

- in Input to the SHARC interface. Driven to the SHARC interface by the PCI core.
- out Output from the SHARC interface. Driven to the PCI core by the SHARC interface.

Typographic Conventions

The following typographic conventions are used in this document:

- *Italic* type is used to designate the following:
  - Document titles (for example, *PCI Local Bus Specification*)
  - Important terms
  - Equation variables

- Signal names are designated as follows: **Signal Name**
- Bit names are designated as follows: *Bit Name*
- Register names are designated as follows: ***Register Name***

Document Status Information

BittWare classifies its technical documentation as either Preliminary or Final.

- *Preliminary*: The Preliminary release of the manual contains information that is subject to change. It contains information about a product that is near production-ready and is revised as required. The Preliminary manual exists until the product is released to production.
- *Final*: The Final manual contains information about a final, customer-ready product. The final manual is available online or as a printed copy.

1.2.3 Revision History

Table 1-1
Document Revision History

Release	Status	Date	Comments
UG-SFIN160G-01	Final	January, 2002	This is the first customer release version of the SFIN-160 *SharcFIN ASIC User's Manual*
UG-SFIN160G-02	Final	December, 2003	Updates to pages 14 and 89: the size of BAR1 is 4 MB rather than 8 MB as stated in the previous version.

1.2.4 Related Documents

Sources for ADSP-21160 Information

For information about the ADSP-21160 SHARC DSP, refer to the following sources:

- *ADSP-21160 SHARC DSP Hardware Reference*
  Analog Devices, Inc.
- *ADSP-21160 SHARC DSP Instruction Set Reference*
  Analog Devices, Inc.

Sources for PCI Information

For information about PCI, refer to the following sources:

- The PCI Special Interest Group provides many resources for PCI information. You can also order copies of the latest PCI specification on their website.
  http://www.pcisig.com
  PCI Special Interest Group
  2575 NE Kathryn St. #17
  Hillsboro, OR 97124

- The most current versions of the PCI Specifications are the following:
  PCI Local Bus Specification, Revision 2.2
  PCI Hot-Plug Specification, Revision 1.1
  PCI Power Management Interface Specification, Revision 1.1

- A good resource if you need an introduction to PCI is:
  PCI System Architecture
  Fourth Addition
  MindShare, Inc.
  Tom Shanley and Don Anderson
  ISBN: 0-201-40993-3

Other Recommended Specifications

Other recommended specifications include the following:

- *PICMG 2.0, Compact PCI Specification, Revision 2.1* (or greater)
  PCI Industrial Computer Manufacturers Group (PICMG)
  401 Edgewater Place, Suite 500
  Wakefield, MA 01880, USA
  Fax: 781-224-1239
  Tel: 781-224-1100
  http://www.picmg.org

- *Intelligent I/O (I_2O) Architecture Specification, Revision 1.5*
  I_20 Special Interest Group
  404 Balboa Street
  San Francisco, CA 94118, USA
  Tel: 415-750-8352
  Fax: 415-751-4829
  http://www.i20sig.org

1.3 Contacting BittWare Technical Support

BittWare is dedicated to providing our customers with superior technical support. We offer the following types of technical support:

- **E-mail:** You can direct questions and feedback to us at support@bittware.com. Please include your name, company, address, phone number, and specific details about your request for technical support.
- **Phone:** You can call us directly at 603.226.0404 between the hours of 8:30 a.m. – 5:30 p.m. Eastern Standard Time.
- **BittWare's website:** Our website at http://www.bittware.com contains the latest manuals, anomaly lists, and FAQs.

Chapter 2
Functional Description

This chapter provides a detailed description of how the SharcFIN ASIC functions. It covers the following topics:

- Using the SharcFIN to access ADSP-21160 DSPs from the PCI host (section 2.1)
- Accessing the SharcFIN from an ADSP-21160 DSP (section 2.2)
- How the SDRAM controller functions and how to configure it (section 2.3)
- The SharcFIN's interrupt and messaging capabilities (section 2.4)
- The peripheral bus interface (section 2.5)
- How to reset the SharcFIN (section 2.6)
- The SharcFIN's DMA and bus mastering operation (section 2.7)
- The I^2C interface (section 2.8)

2.1 PCI-to-DSP Target Interface

This section explains how to use the SharcFIN to access the ADSP-21160 DSPs on the DSP board from the PCI interface. It defines and provides addresses for the base address regions (BARs), describes how the PCI-to-DSP target functions, and describes how to perform target writes.

2.1.1 Overview of the PCI-to-DSP Target

The SharcFIN's PCI-to-DSP target function permits very high data bandwidth for both read and write accesses. As is the case with all master-capable PCI controllers, the target of the SharcFIN is completely independent from the master. It is possible, by design, for any of the SharcFIN's DMA channels to access its own target. The troublesome timing associated when a target accesses its own master is handled internally within the SharcFIN. The SharcFIN accepts multiple target write bursts, but due to transaction ordering requirements, only one non-prefetched burst target read can be pending at a time.

The SharcFIN uses PCI Base Address Regions (BARs) to map its various parts onto the PCI bus. BAR0 maps the SharcFIN's control registers, BAR1 maps to the peripheral bus, BAR2 maps to the ADSP-21160's multiprocessor memory space (MMS), BAR3 is unused, and BAR4 maps to the SDRAM.

The target of the SharcFIN supports three BARs, which support the following features:

- All are memory mapped
  - BAR0 = 512 bytes
  - BAR1 = 4 MB
  - BAR2 = 32 MB
  - BAR4 = 16 MB
- BAR2 and BAR4 are prefetchable, subject to control register settings. Refer to section 3.3.1 for enabling this function.
- BAR2 and BAR4 are 64-bit addressable in systems that support a 64-bit address space on the PCI bus.

2.1.2 How the PCI-to-DSP Target Functions

The PCI bus has access to the control registers of the SharcFIN via the target interface. The control registers are located in BAR0 and occupy the offsets from 0x00–0x5F. Target read access from the control registers is not prefetchable, and neither reads nor writes to the control registers will burst.

BAR0 contains all control registers. Access to these registers from the PCI bus doesn't use the ADSP-21160 cluster bus and will not interfere with transactions occurring on the ADSP-21160 cluster bus. Refer to section 3.3 for a description of these registers.

BAR2 allows access to the MMS space of the ADSP-21160 cluster bus, including all four ADSP-21160 DSPs (on the BittWare Hammerhead DSP board) and the broadcast write space.

BAR4 allows access to a 16 MB window of the SDRAM. A register at location 0x4A in the control registers sets which 16 MB window is currently visible. BAR2 and BAR4 are prefetchable, which means that block reads to these memory regions may be made more efficient by performing burst reads on the ADSP-21160 cluster bus. Prefetching can be enabled for burst reads from the PCI bus or non-burst reads (not all masters will perform burst reads).

BAR2 and BAR4 may also be mapped into 64-bit address space on 64-bit addressable systems, which allows more boards to be mapped into these systems as they can be mapped into areas above the first four gigabytes.

The SharcFIN PCI core supports the following PCI-to-DSP target functions (not including functions implemented in the SHARC interface):

- PCI Rev. 2.2 protocol
- Retries, disconnects, 8/16 **$\overline{\text{TRDY}}$** timeout
- Posts multiple write transactions into the Target Write Post FIFO
- Control register accesses from PCI
- Many other tasks that all PCI devices require

The SHARC interface completes the following steps to make target accesses work:

1. Arbitrates between the PCI core, the ADSP-21160s, and DMA engines for access to its resources.
2. Performs burst accesses to the SDRAM and ADSP-21160s whenever possible.
3. Maintains a low latency, low overhead approach to all transactions.

2.1.3 Performing Target Writes

The memory structure called the Target Write/Post FIFO is dedicated to target write transactions. This FIFO is 72 bits wide by 32 bits deep. All target write operations go through this FIFO, including target writes to the control registers. The FIFO can contain several target write transactions, and the SharcFIN handles transaction ordering issues simultaneously.

Target writes are stacked in this FIFO until it is full. If a target read is pending, the SharcFIN asserts a **$\overline{\text{STOP}}$**, forcing a target-abort-with-retry on all following read transactions until the first read has completed one data phase on PCI. During a target write, an address is tagged and placed into the FIFO, followed by all of the data phases of the current transaction.

The FIFO may contain a range of stacked writes from 16 single data write transactions to a single transaction with 32 quadwords of data. During bursts from a 32-bit master, dwords are packed to quadwords before they are written to the FIFO.

2.1.4 Controlling the FIFO Emptiness Threshold

The SharcFIN contains variables that are used to optimize the control of the Target Write/Post FIFO. Assume the following situation:

> The Target Write/Post FIFO has five remaining quadword locations available, and the PCI wants to burst nine quadwords of data to BAR2

When this transaction is started, the address will be written into the FIFO, followed by four quadwords of data. After the fourth data phase, the SharcFIN will assert a **STOP**, terminating the transaction with a target-abort-with-retry. Since the transaction did not complete fully, the master that initiated the transaction will start another transaction.

At this point, assume that the ADSP-21160 cluster bus has removed three quadwords of data from the FIFO, leaving three locations open. The new address will be written into the FIFO, followed by two quadwords of data. Once again, the SharcFIN will issue a **STOP**, and the cycle will repeat until all nine quadwords of data have been transferred.

This scenario is not optimal since many locations in the FIFO were used to contain address information, and unnecessary PCI bandwidth was used for multiple, separate transactions. It would be better if the SharcFIN simply refused the transaction until enough positions were available in the FIFO to contain the entire transaction. In this case, if the FIFO had ten locations available, the entire transaction would burst, minimizing the bandwidth needed for the PCI bus and maximizing the use of the FIFO.

To help adjust the movement of target data in this scenario, the SharcFIN has a set of variables in the control registers called ***Target FIFO Emptiness Threshold Control***. Each of the four Base Address Regions (BARs) has two bits that control the FIFO threshold at which more data is accepted. If the Target Write/Post FIFO does not contain at least the number of empty quadwords set in this register, then a target write to that particular BAR will be terminated with a target abort. The selectable values for the register are shown in Table 2-1 on page 2-17.

Table 2-1
Number of Quadword Positions That Must Be Empty in the Target Write/Post FIFO for a Target Write to be Accepted

Target FIFO Emptiness Threshold Control	Number of Quadword Positions That Must Be Empty
00	2 or More (1 Address, 1 Data)
01	3 or More (1 Address, 2 Data)
10	Use Threshold set in ***Target FIFO Threshold MSB[3:0]*** register

Register Value	Empty Quadword Positions
0	2 (1 Address, 1 Data)
1	2 (1 Address, 1 Data)
2	4 (1 Address, 3 Data)
3	6 (1 Address, 5 Data)
4	8 (1 Address, 7 Data)
5	10 (1 Address, 9 Data)
6	12 (1 Address, 11 Data)
7	14 (1 Address, 13 Data)
8	16 (1 Address, 15 Data)
9	18 (1 Address, 17 Data)
a	20 (1 Address, 19 Data)
b	22 (1 Address, 21 Data)
c	24 (1 Address, 23 Data)
d	26 (1 Address, 25 Data)
e	28 (1 Address, 27 Data)
f	30 (1 Address, 29 Data)

11	FIFO Must Be Empty

2.2 DSP-to-SharcFIN Target Interface

This section explains how to access the SharcFIN from the ADSP-21160 DSPs.

2.2.1 Overview of the DSP-to-SharcFIN Target

The DSP-to-SharcFIN target consists of three regions of access:

- SDRAM, which maps to ADSP-21160 memory select line MS0
- Devices on the peripheral bus, including the UART and flash, which map to MS1
- SharcFIN control registers, single PCI access, and DMA FIFOs, all of which map to MS2

These three regions of access allow the ADSP-21160s to access all of the functions of the SharcFIN.

2.2.2 How the DSP-to-SharcFIN Target Functions

The SDRAM interface is a synchronous, low-latency, bursting interface. The ADSP-21160s allow burst links of up to four 64-bit words. Latency on these operations for in-page burst access is 2,1,1,1 for writes and 4,1,1,1 for reads. For details on the SDRAM interface, refer to section 2.3.

Access to the peripheral bus is asynchronous, and timing depends on the peripheral that is being accessed. For details on the peripheral bus interface, refer to section 2.5.

The SharcFIN control register, PCI access, and DMA FIFO region is a synchronous interface. Reads and writes to all locations in this space, with the exception of the FIFOs, are 1 wait state operations. For details about the SharcFIN control registers, refer to section 3.3.2; for details about single PCI accesses, refer to section2.7.2; and for details about the DMA FIFOs, refer to section 2.7.3.

2.3 SDRAM Controller

This section explains how the SharcFIN's SDRAM controller functions and describes how to configure it and how to use it to access SDRAM on an ADSP-21160 DSP board. The SDRAM controller supports up to 512 Mbytes of SDRAM. It refreshes the SDRAM and controls all of the interfacing from the ADSP-21160s or the PCI bus to the SDRAM.

2.3.1 Configuring SDRAM

The SharcFIN's SDRAM controller provides an interface to SDRAM on an ADSP-21160 DSP board. It allows access to the SDRAM from either the ADSP-21160 DSPs or from the PCI host. To configure the SharcFIN to allow access to SDRAM, you will need to set three registers:

- The ***SD Size Config*** register sets up the SDRAM for both PCI and ADSP-21160 access
- The ***SDRAM Window*** register sets up the PCI interface to access the SDRAM
- The ADSP-21160s' ***WAIT*** register sets up the ADSP-21160s to access the SDRAM

Setting the SD Size Config Register

The first register to configure to allow access to SDRAM is the ***SD Size Config*** register (address 0x45). This register is a 4-bit wide register. Bits [1:0] determine how the SharcFIN SDRAM controller uses the SharcFIN's **SD_CS0** and **SD_CS1** (chip select 0 and chip select 1) pins (see Table 4-3 on page 4-206 for pin numbers). The chip select pins allow the SharcFIN to be seamlessly connected to more than one bank of SDRAM. These two bits have the following effects:

Table 2-2
Settings for Bits 0 and 1 of the SD Size Config Register

B0	B1	Result
0	0	**SD_CS0** always active
0	1	**SD_CS1** active when 32-bit word address bit 24 is high; otherwise, **SD_CS0** is active
1	0	**SD_CS1** active when 32-bit word address bit 25 is high; otherwise, **SD_CS0** is active
1	1	**SD_CS1** active when 32-bit word address bit 26 is high; otherwise, **SD_CS0** is active

Note *Do not change this register. When you issue a board reset, the DSP21k-SF Toolkit Host Interface Library (HIL) automatically sets this register properly.*

Bit[2] of the ***SD Size Config*** register sets the refresh rate of the SDRAM. Its settings are as follows:

Table 2-3
Settings for Bit 2 of the SD Size Config Register

B2	Result
0	4K refreshes every 64 milliseconds
1	8K refreshes every 64 milliseconds

Bit[3] of the ***SD Size Config*** register resets the SDRAM controller. When a 1 is written to this register, the SDRAM is reset. At the end of reset, the bit is cleared.

Accessing SDRAM From an ADSP-21160 DSP

In the ADSP-21160 memory space, the SDRAM is mapped into MS0, and the ADSP-21160s have full access to all of the SDRAM. To configure the ADSP-21160s for access to the SDRAM, you will need to set the ADSP-21160s' ***WAIT*** register properly. The external bank 0 access mode bit (EB0AM) must be set to synchronous with 1 wait state writes (10), and the external bank 0 wait states (EB0WS) must be set to 1 wait state (001). For specific details on the ***WAIT*** register, refer to the *ADSP-21160 SHARC Hardware Reference Manual*.

Accessing SDRAM From the PCI Interface

From the PCI side, the SDRAM is mapped into BAR4 with a 4 Mword (16 Mbyte) window viewable at a time. Because the SDRAM is so large, this window exists to keep the entire SDRAM from being mapped into PCI memory and consuming an excessive portion of the available PCI address space. The offset into BAR4 provides bits [21:0] of the address of the SDRAM word to be accessed. The contents of the 5-bit ***SD Window*** register (address 0x4A) are prepended to bits [26:22] of the address to complete the address and thereby select the 4 Mword window to be accessed. Bit[31] is always set during PCI-to-SDRAM accesses to indicates to the ADSP-21160s that the transaction is not intended for them.

Using a window register of this type to access the large amount of SDRAM has the following two limitations:

1. Accesses to blocks of data that cross the window boundaries must be handled carefully. The Host Interface Library (in BittWare's DSP21k-SF Toolkit) takes care of this limitation for host-based transactions that use its functions to access the SDRAM. Attempting to access a block of data that crosses a window boundary would require breaking the transfer into two smaller transfers and handling the window register properly for each transfer. Non HIL-based transfers, such as bus mastering board-to-board transfers must handle this situation.
2. The window register is a shared resource. Because the SharcFIN uses the window register for every PCI access to SDRAM, be careful to coordinate SDRAM accesses from PCI if you have multiple threads on the host or multiple PCI bus masters accessing the SDRAM. More than one device attempting to access different windows of the SDRAM

simultaneously will require great care in keeping the SDRAM window register set correctly for each access.

In addition to these two limitations, the manner in which the ADSP-21160s address the SDRAM causes some confusion about how the pages are ordered. MS0 on the ADSP-21160 starts at 0x800000. Therefore for the PCI bus to access the start of SDRAM from the ADSP-21160's perspective, you will need to set the window bits to 0x02 so that the cluster bus address is 0x800000. This is made more confusing because the last two 16 MByte blocks of SDRAM as seen from the ADSP-21160s map to window register values of 0x00 and 0x01. These are always the last two, regardless of the SDRAM size. Therefore, window register 0x02 is the first 16 MByte block of SDRAM, 0x03 is the second, 0x04 is the third, ... , 0x00 is the second last, and 0x01 is the last. After reset, the default value of the ***SD Window*** register is 0x02.

2.3.2 SDRAM Timing

Timing From the ADSP-21160

SDRAM timing from the ADSP-21160 is synchronous, 1 wait state. A single write access takes two bus cycles. Since each additional write is single cycle, using the ADSP-21160's burst mode, you can achieve a four 64-bit word burst write in five bus cycles. Reads require additional setup in the SDRAM, resulting in four bus cycles for the first access and a four 64-bit word burst read in seven cycles. The SharcFIN SDRAM controller has a fixed page size of 512 words. Because the SDRAM is page based, you will encounter additional latencies when page boundaries are crossed.

To achieve optimal system performance, use the power of the ADSP-21160's DMA engines and its dual ported internal RAM. Using these features, you can perform DMA-based SDRAM accesses at the same time that the ADSP-21160 core is performing processing on its internal data space.

Timing from the PCI Interface

Timing for access to SDRAM from the PCI interface is similar to timing from the ADSP-21160 with the following two exceptions:

1. Bursts from the PCI bus are limited to the full page (512 words) instead of the four 64-bit words allowed by the ADSP-21160s.
2. An additional latency is imposed on reads to the SDRAM as they propagate through the SharcFIN.

2.4 Interrupts and Messaging

This section discusses the SharcFIN's interrupt capabilities. The SharcFIN's interrupt functionality is broken into two parts. The first is the PCI interrupt controller, which controls the interrupts to and from the PCI interface. The second is the SHARC interface interrupt multiplexer, which routes interrupts to the ADSP-21160s, the PCI interface, or an attached PMC+ card from a variety of sources.

This explains how to use and configure the PCI side interrupt controller and the SHARC interface interrupt multiplexer to generate interrupts to and from the PCI interface, the ADSP-21160 DSPs, and various other on-board peripherals, such as a UART or a PMC card. It also discusses how to determine the status of interrupts.

2.4.1 PCI Interrupt Controller and Status

The SharcFIN PCI core contains a programmable interrupt controller that can be configured to generate an interrupt on any or all of the events in "Determining the Interrupt Status" on page 22. Two separate interrupts can be generated: one for the PCI bus, and one for the ADSP-21160 cluster bus.

The interrupt signal on the PCI bus is named **$\overline{\text{INTA}}$**. The interrupt signal to the SharcFIN's interrupt multiplexer is called **PCFlg**. The signal **PCFLg** is synchronous to **LClk** and may be processed in the SHARC interface interrupt multiplexer to create a level or edge style signal of any level or duration.

Each interrupt source can be masked separately for each interrupt, and the status of all interrupt sources can be read from both the PCI bus and the ADSP-21160 cluster bus. See the descriptions for control register 0x22 (Table 3-108 on page 3-147) for the mask locations.

Once an interrupt has been generated, you can determine the source of the interrupt by reading the interrupt status register. In the case of I_2O, two reads are necessary since determining the source of potential interrupts spans two control registers, 0x20 and 0x0C. The location of the registers for the PCI bus is offset 0x20 from ***Base Address Register 0*** (BAR0), bits [56:32]. For the SHARC interface, the registers are accessed via the ADSP-21160 cluster bus and are at offset 0x20, bits [27:0].

Each interrupt signal has its own set of masks called *interrupt masks*. These bits are located at 0x22 in the control registers. When a mask bit for a particular condition is set to '1', the respective interrupt will go active when the event or action occurs.

Each interrupt signal has its own set of status bits called *interrupt status*. These bits are located at 0x20 in the control registers. An interrupt will stay active until the source(s) of the interrupt is/are cleared. An interrupt is cleared by writing a '1' to the corresponding bit in the ***Interrupt Status*** register (0x20). A write of '0' to a bit in the ***Interrupt Status*** register has no effect.

Determining the Interrupt Status

The SharcFIN contains a status register that allows you to determine the state or status of PCI related operations or actions. The ***Status*** register is at offset 0x24 and can be read by

either the host (via PCI) or by the ADSP-21160s (via the ADSP-21160 cluster bus). The host controller gets this status information by reading the lower four byte lanes of this register. The ADSP-21160 cluster bus gets this status information by reading the lower four byte lanes of this register. All bits are set and cleared automatically by the PCI controller with the exception of the DMA start/done bits. Table 2-4 below describes the contents and settings of the ***Status*** register.

Table 2-4
Contents and Settings of the Status Register (0x24)

Bit	Name	Description
7:0	USER Incoming/PCI Outgoing Mailbox Status	Each bit represents the empty/full status of respective byte lanes of the mailbox register 0x1C. • A bit is set when a PCI write occurs to a byte lane of mailbox register 0x1C. • A bit is cleared when a read from the respective byte lane of the mailbox is done from the ADSP-21160 cluster bus.
15:8	PCI Incoming/USER Outgoing Mailbox Status	Each bit represents the empty/full status of respective byte lanes of the mailbox register 0x1E. • A bit is set when an ADSP-21160 cluster bus write occurs to a byte lane of mailbox register 0x1E. • A bit is cleared when a read from the respective byte lane of the mailbox is done from PCI.
20:16	DMA Start/Done# Status	Each bit represents the running/stopped status of each of the five DMA controllers. The five channels are Transmit 0/1, Receive 0/1, and Chaining. • A bit is set when a DMA operation is started. A DMA operation is started by a write of '1' to the respective bit. • A bit will be cleared automatically when the DMA operation is complete and the DMA channel is idle. • A write of '1' to a channel that is currently running has no effect. • A write of '0' has no effect at any time.
21	Chain Fetch End Status	This bit indicates whether or not the last chain descriptor in a chain has been fetched. • The bit is set when the last chain descriptor in a chain has been fetched. • The bit is cleared when a chain operation is started.

Table 2-4
Contents and Settings of the Status Register (0x24) (Continued)

Bit	Name	Description
23	BIST Start/Done	This describes the execution status of the Build-In-Self-Test (BIST) mechanism. • The bit is set when BIST is started by a write of '1' to the BIST configuration bit in the PCI config space (0x03, bits 31:24]). • The bit is cleared by an ADSP-21160 cluster bus write to bit[12] of control register 0x34.
24	I_2O Inbound Free List Empty	• This bit is set when the PCI host reads from the inbound queue pointer (0x10, bits [31:0]) such as fetching a pointer to a free message frame. • The bit is cleared when the ADSP-21160 cluster bus writes to the inbound queue pointer (0x10, bits [31:0]).
25	I_2O Inbound Post List Full	• This bit is set when the PCI host writes to the inbound queue pointer (0x10, bits [31:0]) such as placing a message frame in the queue. • The bit is cleared when the ADSP-21160 cluster bus reads from the inbound queue pointer (0x10, bits[31:0]).
26	I_2O Outbound Post List Empty	• This bit is set when the PCI host reads from the outbound queue pointer (0x10, bits [63:32]) such as fetching an outbound frame pointer. • The bit is cleared when the ADSP-21160 cluster bus writes to the outbound queue pointer (0x10, bits [63:32]).
27	I_2O Outbound Free List Full	• This bit is set when the PCI host writes to the outbound queue pointer (0x10, bits [63:32]) such as returning a free frame to the list. • The bit is cleared when the ADSP-21160 cluster bus reads from the outbound queue pointer (0x10, bits [63:32]).

Configuring the Interrupt Sources

This section indicates the possible sources for interrupts to the SharcFIN's PCI core or the interrupt multiplexer.

Note *In the following descriptions, these conventions apply:*

- *The term "host" refers to a controlling processor that responds to the PCI interrupts via the PCI bus.*
- *The term "user" refers to a local processor connected to the SHARC interface that responds to the ADSP-21160 interrupt via the ADSP-21160 cluster bus.*

The interrupt signals **$\overline{\text{INTA}}$** (host) or **PCFlg** (SHARC interface) can be asserted by one or more of the following conditions:

Chain Fetch Done
The last Chain Descriptor Table has been fetched, indicating that it is now okay to reclaim the memory used for the tables.

Chain Done
The last chain descriptor has finished executing and the chaining DMA controller is now idle.

DMA Receive Channel 1 Done
The DMA Receive Channel 1 controller has completed all the requested transfers and is now idle.

DMA Receive Channel 0 Done
The DMA Receive Channel 0 controller has completed all the requested transfers and is now idle.

DMA Transmit Channel 1 Done
The DMA Transmit Channel 1 controller has completed all the requested transfers and is now idle.

DMA Transmit Channel 0 Done
The DMA Transmit Channel 0 controller has completed all the requested transfers and is now idle.

One or more of the following conditions can cause the assertion of the signal **PCFlg** (SHARC interface):

I_2O Outbound Free List Full
This interrupt is asserted when the PCI host writes to the outbound queue pointer, such as returning a free frame to the list.

I_2O Outbound Post List Empty
This interrupt is asserted when PCI reads from the outbound queue pointer, such as fetching an outbound frame pointer.

I_2O Inbound Post List Full
This interrupt is asserted when PCI writes to the inbound queue pointer, such as placing a message frame in the queue.

I_2O Bound Free List Empty
This interrupt is asserted when PCI reads from the inbound queue pointer, such as fetching a pointer to a free message frame.

BIST Start
The host has written a "1" to the BIST configuration register, requesting that the built-in-self-test (BIST) function be executed and results reported.

Single PCI Access Complete
The single PCI access operation has successfully completed.

User Outgoing Mailbox Byte 7 Empty
The host has read mailbox byte 7 and this mailbox is now empty.

User Outgoing Mailbox Byte 6 Empty
The host has read mailbox byte 6 and this mailbox is now empty.

User Outgoing Mailbox Byte 5 Empty
The host has read mailbox byte 5 and this mailbox is now empty.

User Outgoing Mailbox Byte 4 Empty
The host has read mailbox byte 4 and this mailbox is now empty.

User Outgoing Mailbox Byte 3 Empty
The host has read mailbox byte 3 and this mailbox is now empty.

User Outgoing Mailbox Byte 2 Empty
The host has read mailbox byte 2 and this mailbox is now empty.

User Outgoing Mailbox Byte 1 Empty
The host has read mailbox byte 1 and this mailbox is now empty.

User Outgoing Mailbox Byte 0 Empty
The host has read mailbox byte 0 and this mailbox is now empty.

User Incoming Mailbox Byte 7 Full
The host has filled mailbox byte 7 and can be read by the user.

User Incoming Mailbox Byte 6 Full
The host has filled mailbox byte 6 and can be read by the user.

User Incoming Mailbox Byte 5 Full
The host has filled mailbox byte 5 and the mailbox can be read by the user.

User Incoming Mailbox Byte 4 Full
The host has filled mailbox byte 4 and the mailbox can be read by the user.

User Incoming Mailbox Byte 3 Full
The host has filled mailbox byte 3 and the mailbox can be read by the user.

User Incoming Mailbox Byte 2 Full
The host has filled mailbox byte 2 and the mailbox can be read by the user.

User Incoming Mailbox Byte 1 Full
The host has filled mailbox byte 1 and the mailbox can be read by the user.

User Incoming Mailbox Byte 0 Full
The host has filled mailbox byte 0 and the mailbox can be read by the user.

The signal **$\overline{\text{INTA}}$** can also be asserted by one or more of the following conditions:

Signal PCFlg Asserted (User Interrupt)
The ADSP-21160 cluster bus has asserted the signal **PCFlg**.

PCI Incoming Mailbox Byte 7 Full
The user outgoing mailbox byte 7 has been filled and can be read by the host.

PCI Incoming Mailbox Byte 6 Full
The user outgoing mailbox byte 6 has been filled and can be read by the host.

PCI Incoming Mailbox Byte 5 Full
The user outgoing mailbox byte 5 has been filled and can be read by the host.

PCI Incoming Mailbox Byte 4 Full
The user outgoing mailbox byte 4 has been filled and can be read by the host.

PCI Incoming Mailbox Byte 3 Full
The user outgoing mailbox byte 3 has been filled and can be read by the host.

PCI Incoming Mailbox Byte 2 Full
The user outgoing mailbox byte 2 has been filled and can be read by the host.

PCI Incoming Mailbox Byte 1 Full
The user outgoing mailbox byte 1 has been filled and can be read by the host.

PCI Incoming Mailbox Byte 0 Full
The user outgoing mailbox byte 0 has been filled and can be read by the host.

PCI Outgoing Mailbox Byte 7 Empty
The user has read the PCI outgoing mailbox byte 7 and this mailbox is empty.

PCI Outgoing Mailbox Byte 6 Empty
The user has read the PCI outgoing mailbox byte 6 and this mailbox is empty.

PCI Outgoing Mailbox Byte 5 Empty
The user has read the PCI outgoing mailbox byte 5 and this mailbox is empty.

PCI Outgoing Mailbox Byte 4 Empty
The user has read the PCI outgoing mailbox byte 4 and this mailbox is empty.

PCI Outgoing Mailbox Byte 3 Empty
The user has read the PCI outgoing mailbox byte 3 and this mailbox is empty.

PCI Outgoing Mailbox Byte 2 Empty
The user has read the PCI outgoing mailbox byte 2 and this mailbox is empty.

PCI Outgoing Mailbox Byte1 Empty
The user has read the PCI outgoing mailbox byte 1 and this mailbox is empty.

PCI Outgoing Mailbox Byte 0 Empty
The user has read the PCI outgoing mailbox byte 0 and this mailbox is empty.

Determining the Interrupt Timing

Assuming the correct masks are asserted, when a DMA operation has completed, the interrupts will go active according to the following approximate timing:

PCFlg: 12 clock cycles of **LClk** or **PCI_clk** (whichever is slower)+ 2 clock cycles of **LClk**

$\overline{\text{INTA}}$: 12 clock cycles of **LClk** or **PCI_clk** (whichever is slower) + 2 clock cycles of **LClk** + 3 clock cycles of **PCI_clk**

The interrupts will deassert according to the following approximate timing:

PCFlg: 2 clock cycles of **LClk**

$\overline{\text{INTA}}$: Unknown because the time for the write to complete is indeterminate.

Interrupt Service Routine

The deassertion time after a write of "1" to a bit of the interrupt status register for **$\overline{\text{INTA}}$** is unknown, which is troublesome for the interrupt service routine on the host processor. It is impossible to determine the latency of a PCI write, since bridges and other such devices could be between the host processor and the SharcFIN. To guarantee that an **$\overline{\text{INTA}}$** interrupt is deasserted, the interrupt service routine for the host processor should do a write of "1" to the correct bits of the ***interrupt status*** register of the SharcFIN, followed by a read of any register in the SharcFIN before the routine is exited.

Many devices can be between the SharcFIN and the host processor, and each device in the chain will add some latency to this transaction. The write to clear the interrupt, therefore, could take a long time to execute. This execution time will depend on the following:

- The traffic on the PCI bus(es)
- The number of P2P bridges between the host processor and the SharcFIN
- The number of target transactions stacked in the Target FIFO of the SharcFIN.

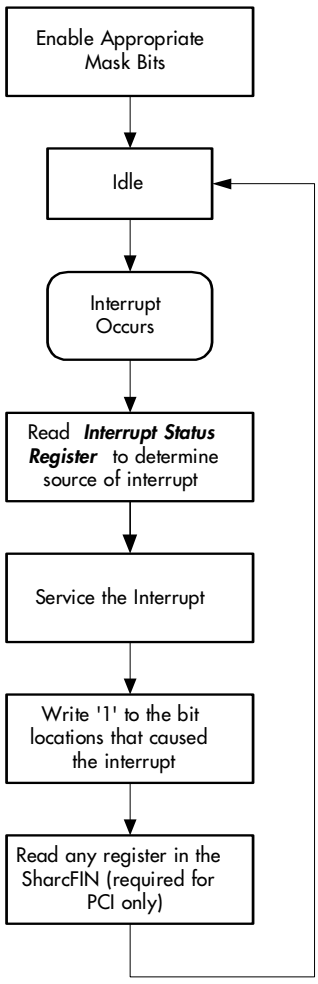
Also, the bridges between the host processor and the SharcFIN will slow down the **$\overline{\text{INTA}}$** signal on its path back to the host processor.

If this write-followed-by-read sequence is not followed, it is possible that the write intended to clear the interrupt will not have executed before the interrupt service routine is exited, and the host processor will go back into the interrupt service routine because **INTA** will still be active. The only way to guarantee that this problem is avoided is to execute a read access to the same device that is generating the interrupt. PCI transaction ordering requirements force the write access to complete at the SharcFIN before the read, guaranteeing execution of the interrupt-clearing write.

The read access will also take a longer path back to the host processor than the **INTA** signal (by PCI specification), and thereby guarantee that the host processor will not exit the interrupt routine before the **INTA** signal has been deasserted. On the ADSP-21160 side, the read is not required since a fixed number of clock cycles exists between the time of the write and the deassertion of **PCFlg**.

The PCI interrupt **INTA** is synchronized to the PCI clock, **PCI_clk**. The ADSP-21160 side interrupt **PCFlg** is synchronized to the PCI clock, **LClk**. A flow chart that illustrates the steps is shown in Figure 2-1.

Figure 2-1
Interrupt Service Routine Flow Chart



2.4.2 SHARC Interface Interrupt Multiplexer

The SharcFIN contains a flexible interrupt multiplexer that you can use to create complex interrupt schemes on an ADSP-21160 DSP board. The interrupt multiplexer contains an interrupt multiplexer for each ADSP-21160 on the DSP board, the PCI interface, and any PMC+ cards attached to the DSP board. Inputs to the multiplexer are flags from each ADSP-21160, a PCI side flag, PMC+ flags, UART flags, a peripheral bus flag, and a DMA flag. Outputs from the multiplexer are an interrupt line to each ADSP-21160 on the DSP board, the PCI side, and a PMC+ site on the DSP board, if applicable.

How the Interrupt Multiplexer Functions

The interrupt multiplexer for each output is completely independent and can handle multiple input sources. Input sources include a flag from each of the four ADSP-21160s, a flag from the PMC+ interface, a PCI flag, two UART flags, a peripheral bus flag, and a DMA flag. A second flag from each ADSP-21160 also functions as an input to the multiplexer but can only be used to generate interrupts to other ADSP-21160s.

Each interrupt multiplexer consists of a 32-bit configuration register that allows you to select the desired interrupt sources (see “Interrupt Configuration Registers” on page 188). Each register is broken into two parts. Bits [15:0] are read/write mask bits that select from the sources that will be generating the interrupt; a 1 in a bit position enables the corresponding flag to cause an interrupt on that multiplexer. Bits [31:16] are read only masked flag lines; when one of the flag inputs and its corresponding bit in bits [15:0] of the configuration register is high, the corresponding bit in bits [31:16] is also set to indicate the source of the interrupt. Bits [31:16] allow a device that receives an interrupt to quickly determine the source of the interrupt.

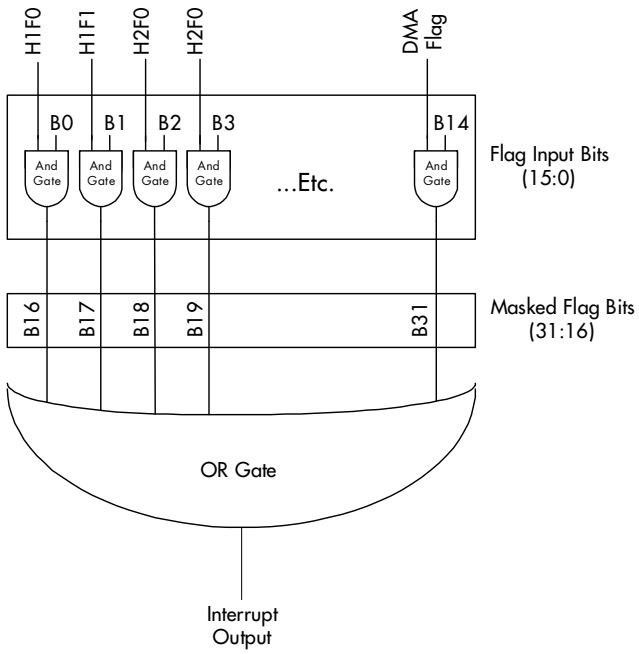
To generate an interrupt, both the flag input from the desired source and its corresponding bit in the configuration register must be asserted. Any other flag input and its corresponding bit in the register can also be high to generate an interrupt. The interrupt multiplexer ANDs each flag input and its corresponding mask bit; it then ORs the results of the AND operations together to create the output. Figure 2-2 on page 2–31 illustrates an interrupt configuration register.

Note *The interrupt multiplexer is level sensitive and does not latch interrupt sources. Therefore, the interrupt is active as long as the source is asserted.*

Configuring the Interrupt Sources

To configure the source of an interrupt, you will need to set the appropriate bit in the interrupt configuration register of the desired interrupt output multiplexer. When the flag input that corresponds to that bit is also high, the interrupt multiplexer will generate an interrupt.

Figure 2-2
Interrupt Configuration Register



Performing PCI Side Interrupts

The SharcFIN provides full I^2O support with the associated mailboxes. To generate interrupts from the PCI side, either write to PCI outgoing mailboxes or use the ADSP-21160 side PCI interrupt multiplexer, which generates the PCI side user interrupt bit. To generate interrupts to the PCI side, configure the multiplexer, which will generate the "user side" flag into the PCI side interrupt mechanism. The PCI side must then "open" the interrupt.

Performing ADSP-21160 Interrupts

The PCI side can interrupt the ADSP-21160s by writing into mailboxes. PCI side interrupts into the SharcFIN via the I^2O mailbox registers show up as PCI flags into the SharcFIN interrupt multiplexer. Therefore, you can program the ADSP-21160s to respond to PCI interrupts as desired. Writing into mailboxes will cause the PCI interrupt bit to be set in the SHARC interface interrupt multiplexers, which will generate an ADSP-21160 interrupt if the mask is open.

Determining the Status of the Interrupts

The registers at offset 0x5E and 0x5F show the status of each of the flags and interrupts. The ***Flag Status*** register (0x5E) shows the state of all flag inputs to the interrupt multiplexer. The ***Interrupt Status*** register (0x5F) shows the state of all interrupt outputs from the interrupt multiplexer.

2.4.3 ADSP-21160 Flag and Interrupt Connections to SharcFIN

Each ADSP-21160 SHARC DSP has two flags and two interrupts that connect to the SharcFIN. Flag0, Flag1, and IRQ0 from each DSP connect to the SharcFIN ASIC interrupt multiplexer (see section 2.4.2). IRQ1 from each DSP is hardwired (non-configurable) in the SharcFIN with the following connections:

- 21160-1 IRQ1 is hardwired to 21160-3 Flag1 and 21160-4 Flag1.
- 21160-2 IRQ1 is hardwired to 21160-3 Flag1 and 21160-4 Flag1.
- 21160-3 IRQ1 is hardwired to 21160-1 Flag1 and 21160-2 Flag1.
- 21160-4 IRQ1 is hardwired to 21160-1 Flag1 and 21160-2 Flag1.

2.4.4 Messaging

Mailbox Operation

Functional Description. The SharcFIN has two 64-bit mailbox registers, which are useful for passing command and status information between the ADSP-21160 cluster bus and the PCI bus. The mailbox registers are divided into two 8-byte sets. Each set is dedicated to one bus for transfer of data to the other bus. The mailbox registers at offset 0x1C are ***PCI Outgoing*** or ***User Incoming*** registers. The registers at offset 0x1E are ***User Outgoing*** or ***PCI Incoming*** registers. Figure 2-3 shows a block diagram of the mailbox section of the SharcFIN.

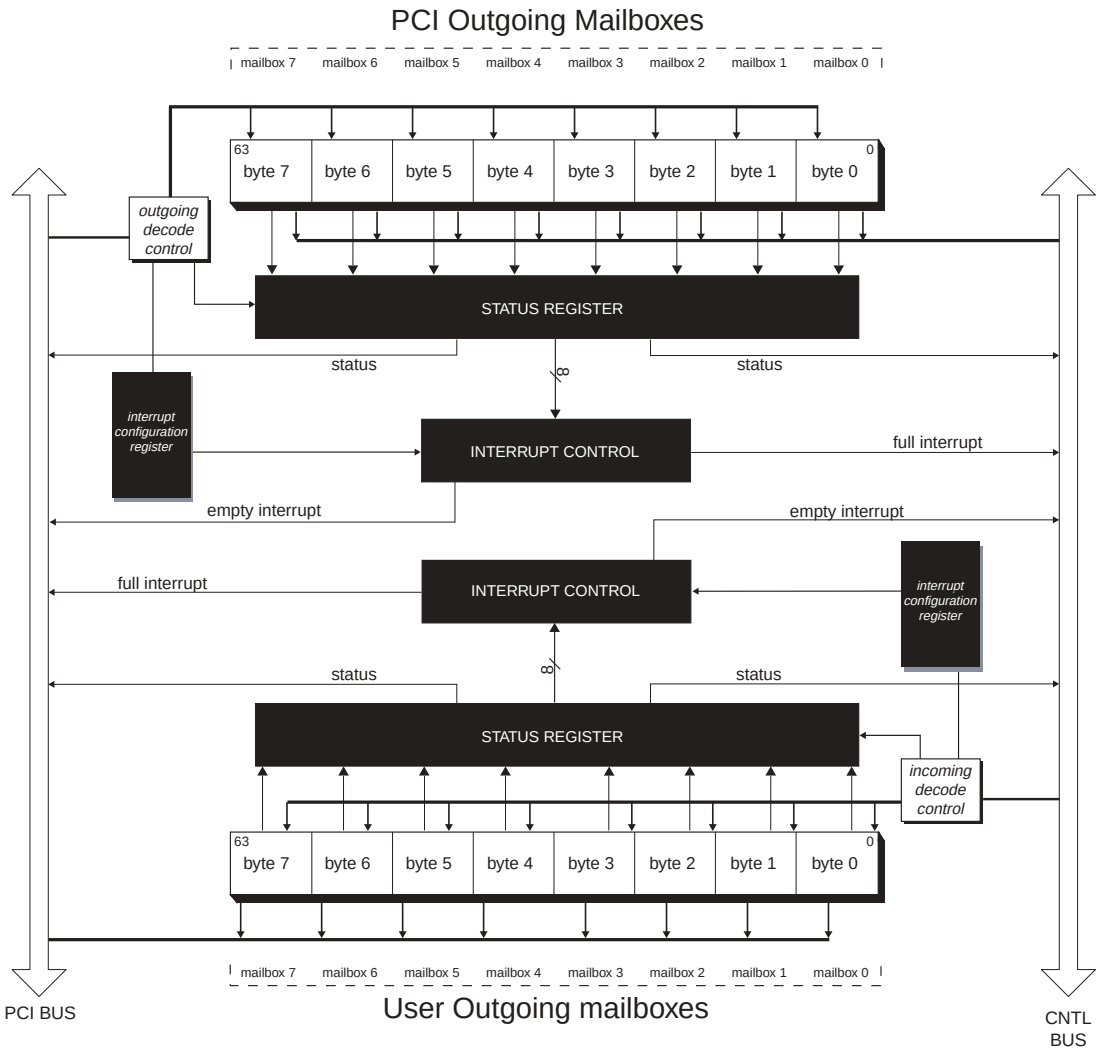
Each byte lane of the mailbox registers contains a status register. Empty and full status is determined by polling the status registers accessible to both the PCI and ADSP-21160 cluster busses. The status registers are accessible to the PCI bus from the control registers (***BAR0)*** at offset 0x24, bits [15:0]. The ADSP-21160 cluster bus accesses the mailbox register at offset 0x24 bits [15:0]. Any of the empty or full flags may be used to generate a PCI or ADSP-21160 interrupt.

Each byte lane of the mailbox registers also contains an empty interrupt flag and a full interrupt flag. Any of these flags may be used to generate a PCI (**$\overline{\text{INTA}}$**) or SHARC interface (**PCFlg**) interrupt. There is a mask register for each of these flags. For example, the ***PCI Outgoing/User Incoming*** mailbox register 7 (0x1C, bits [63:56]) has a *User Incoming MB Full Interrupt Flag 7* (0x20, bit[7], accessible only from the ADSP-21160 cluster bus) and a *PCI Outgoing MB Empty Interrupt Flag 7* (0x20, bit[39], accessible only from PCI). The *User Incoming MB Full Interrupt Mask 7* (0x22, bit[7]) determines if an interrupt (on **PCFlg** only) will be generated from the *User Incoming MB Full Interrupt Flag 7* being set, and the *PCI Outgoing MB Empty Interrupt Mask 7* (0x22, bit[39]) controls whether an interrupt (on **$\overline{\text{INTA}}$** only) will be generated from the *PCI Outgoing MB Empty Interrupt Flag 7* being set.

Mailbox Empty/Full Conditions. The PCI bus and the ADSP-21160 cluster bus each have a mailbox register indicating the empty/full status of each byte lane. A write to an empty outgoing mailbox sets the status bit for that register, marking the mailbox as full. A read from a full mailbox clears the status bit for that register.

The status of a mailbox must be determined prior to filling it. If a full mailbox is written to, the new data overwrites the old, and the mailbox remains full. A read from an empty mailbox returns the previously written data, and the mailbox remains empty. For example, a PCI write to a ***PCI Outgoing/User Incoming*** mailbox register will set the corresponding *User Incoming Full Interrupt Flag*. An interrupt may be generated, and a read from the ADSP-21160 cluster bus to this register also sets the *PCI Outgoing Empty Interrupt Flag*, but it does not clear the *User Incoming Full Interrupt Flag*. These flags can only be cleared by writing a “1” to them from their respective sides.

Figure 2-3
SharcFIN Mailbox Block Diagram



Interrupts. The user has the option to generate interrupts to PCI (using $\overline{\text{INTA}}$) and the ADSP-21160 cluster bus (via the SHARC interface interrupt multiplexer using **PCFlg**) when specific mailbox events occur. The following two conditions, if enabled, will generate an interrupt:

- An incoming mailbox becomes full
- An outgoing mailbox becomes empty

Multiple PCI Masters. The PCI bus has some shortcomings when multiple masters are attempting to read or write the same address location. To maintain proper transaction

ordering as required by the PCI Specification, the SharcFIN must execute all PCI writes that it has received before servicing a target read. Reads and writes to the mailbox registers are target accesses.

Since the SharcFIN can buffer target writes, it may not be able to respond to a target mailbox read within 16 **pci_clk_2fpga's** from the assertion of the **$\overline{\text{FRAME}}$**. Under these conditions, the PCI Specification requires that the SharcFIN terminate the read transaction with a target retry. If the PCI bus is arbitrated to another master, and this different master attempts to read the same target address, the new master will get the data from the prior transaction. This problem is endemic to PCI. To avoid it, use one of the following workarounds:

- Limit the number of masters that access the mailbox locations to two. Assign the first master the mailboxes 3–0 and assign the second master the mailboxes 7–4. Restrict each master to dword assesses only. Since the dword addresses are different, the problem never occurs.
- Add additional mailboxes at different addresses in the SHARC interface using the target interface. Alias the same mailboxes to different target addresses so that each master has a different address.

Intelligent I/O (I_2O)

The I_2O-compatible Messaging Unit supplies two paths for messages: two inbound FIFOs to receive messages from the primary PCI Bus, and two outbound FIFOs to pass messages to the primary PCI Bus. Refer to *I_2O Architecture Specification v1.5* for details. Figure 2-4 and Figure 2-5 illustrate information about I_2O architecture.

Figure 2-4
Typical I_2O Server/Adapter Card Design

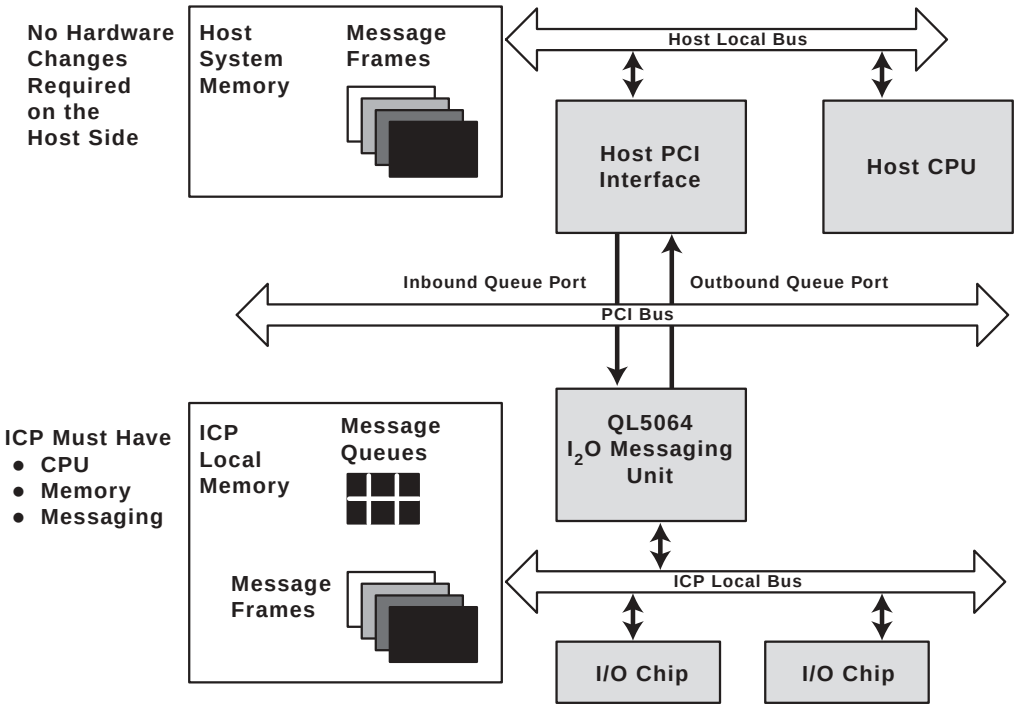
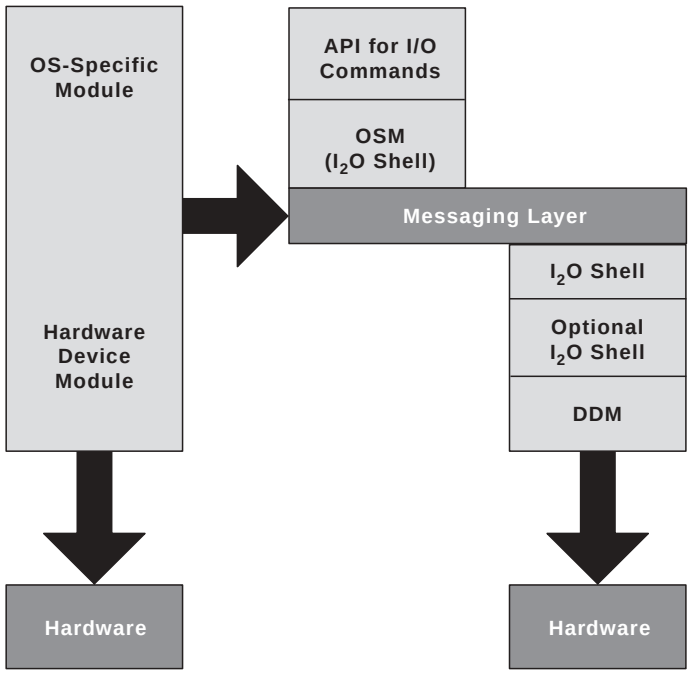


Figure 2-5
Driver Architecture Compared



2.5 Peripheral Bus Interface

The peripheral bus is a general-purpose utility bus that allows easy interface to standard asynchronous peripherals such as UARTs and flash memory. It provides a simple, glueless way to add additional functionality to your ADSP-21160 DSP board. The peripheral bus is an 8-bit wide, 2 MB addressable bus with select lines for three different devices. It is mapped into the ADSP-21160 space as MS1 and into PCI space as BAR1. This region is also mapped into the ADSP-21160 BMS (Boot Memory Select) space to allow the ADSP-21160s to boot from flash memory.

2.5.1 Configuring the Peripheral Bus

Configuring the ADSP-21160s to Access the Peripheral Bus

To configure the ADSP-21160s for access to the peripheral bus, the ADSP-21160s' ***Wait Mode*** register must be set properly. The *external bank 1 access mode* bits must be set to 00 (asynchronous mode). The *external bank 1 wait state* bit must also be set appropriately for the device that the ADSP-21160 is attempting to communicate with. For more information on the ***Wait Mode*** register, refer to the *ADSP-21160 SHARC DSP Hardware Reference Manual* (available from Analog Devices).

Configuring the PCI Bus to Access the Peripheral Bus

The PCI bus accesses the peripheral bus through BAR1. A control register at 0x42, the ***Peripheral Bus Configuration*** register, sets the timing of PCI access to the peripheral bus. It is similar in function to the wait state setting for the ADSP-21160s.

Bits [3:0] of the ***Peripheral Bus Configuration*** register set the length of the transaction. The value indicates the number of local bus clock cycles the transaction will remain active, which allows communication with slower asynchronous devices. For example, with the default count of 5 and a 40 MHz local clock, a write to the peripheral bus holds the peripheral bus write pin, address pins, and data pins valid for 5×25 ns (see Table 4-2 on page 4-200 for pin descriptions). This applies only to PCI transactions.

Setting bit [4] of the ***Peripheral Bus Configuration*** register enables the **PR_ACK** pin to hold off completion of the transaction at the end of the wait programmed by bits [3:0]. Bit [4] is cleared by default. Be cautious about enabling this bit; long hold-offs on the PCI bus can cause your system to become unstable.

Setting bit [5] of the ***Peripheral Bus Configuration*** register resets devices on the peripheral bus. Devices on the peripheral bus are held in reset until the bit is cleared.

2.5.2 Data Alignment on the Peripheral Bus

ADSP-21160 Access to the Peripheral Bus

When an ADSP-21160 attempts to write to the peripheral bus, data must appear on bits [8:0] of the ADSP-21160 cluster bus on even address accesses and on bits [39:32] on odd word accesses. On reads of the peripheral bus by the ADSP-21160, the data appears on bits [8:0] and bits [39:32]. Therefore, when the ADSP-21160 cluster bus is configured as a 64-bit bus with no external port packing, data to/from the devices appears as the least significant byte of the word being accessed.

PCI Access to the Peripheral Bus

For PCI accesses to the peripheral bus, all transactions must be single-byte operations. Never read or write a whole 32-bit word – just a byte. This PCI region of memory is byte mapped instead of word mapped (BAR0, BAR2, BAR4 are all word mapped).

2.5.3 Mappings of Peripherals on the Peripheral Bus

The SharcFIN has select lines for up to three devices on the peripheral bus. The **PR_FLASHSel** line is intended to be connected to flash memory for flash boot or non-volatile storage for the ADSP-21160s. The **PR_UARTSel** line is intended to be connected to a National Semiconductor 16552 dual UART chip. The **PR_Sel** line is intended for custom user applications. The different select lines are accessed as described in Table 2-5 below.

Table 2-5
Memory Mappings of Peripherals on the Peripheral Bus

ADSP-21160 Mappings		PCI Mappings	
Device	**Address Range**	**Device**	**Address Range**
Flash	BMS + 0x0 to 0x1FFFFF		
Flash	MS1 base + 0x0 to 0x1FFFFF	Flash	BAR1 base + 0x0 to 0x1FFFFF
UART	MS1 base + 0x200000 to 0x20000F repeated over and over until 0x3FFFFF	UART	BAR1 base + 0x200000 to 0x20000F repeated over and over until 0x2FFFFF
PB Con*	MS1 base + 0x400000 to 0x5FFFFF	PB Con	BAR1 base + 0x300000 to 0x3FFFFF (B20 is lost)
No Sel†	MS1 base + 0x600000 to 0x7FFFFF		

* PB Con is the peripheral bus connector select.
† For this region, the transaction shows up on the peripheral bus read and write lines, and the address lines are strobed, but no selects are active.

Refer to the documentation for the UART and the flash memory for details on how they function. BittWare also provides example software for using the UART and the flash memory.

2.6 Reset and Watchdog Interface

2.6.1 SharcFIN Reset Sources

The SharcFIN has multiple reset sources that affect different portions of the SharcFIN and devices attached to it. The first three reset sources come from the PCI portion of the chip:

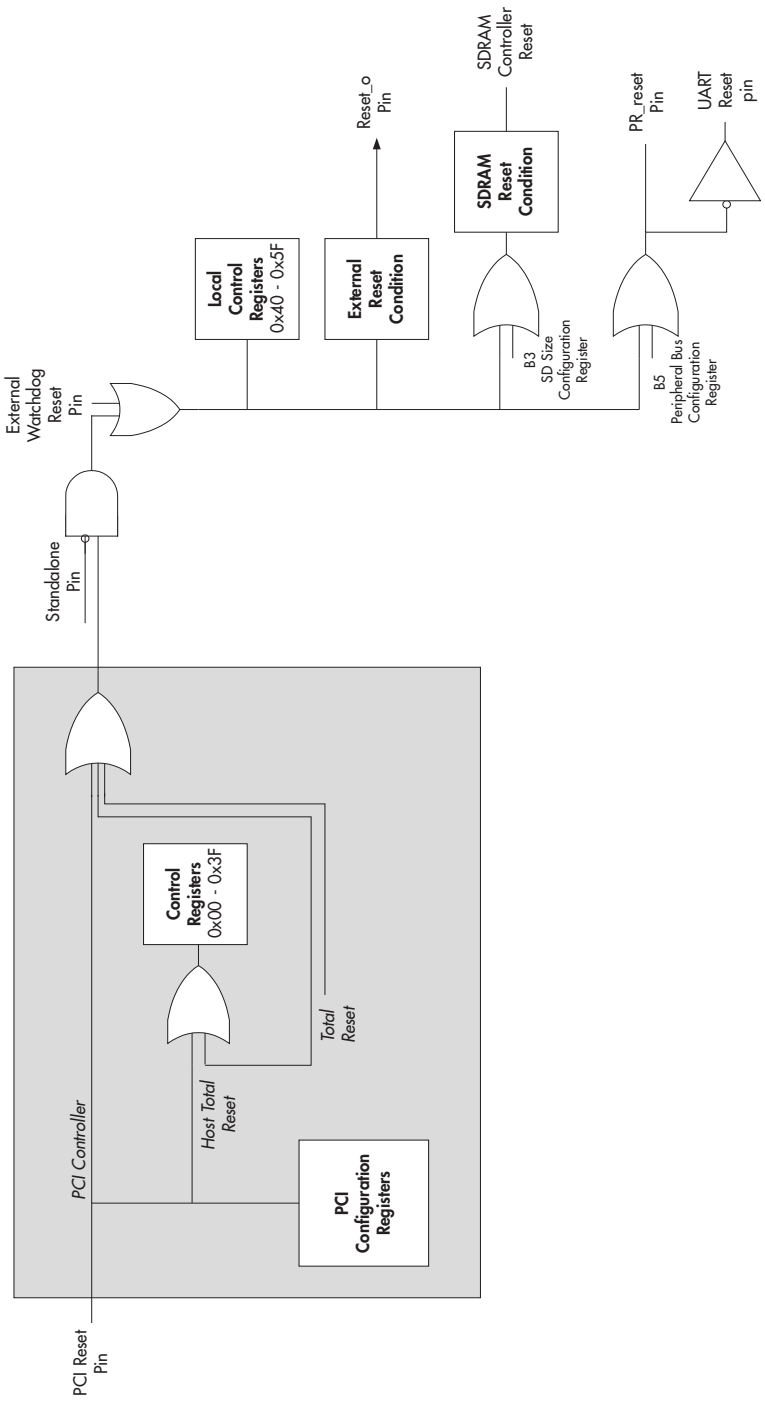
- Total Reset: Resets the entire chip and all peripherals attached to it
- Host Total Reset: Resets the SHARC interface, its peripherals, and the control registers but does not reset the PCI configuration registers
- SHARC interface Reset: Resets the SHARC interface but does not reset the PCI core

The remaining reset sources affect only the SHARC interface portion of the chip and its peripherals:

- External Reset Line: Resets the SHARC interface and all peripherals
- SDRAM Reset Line: Resets the SDRAM controller
- Peripheral Bus Reset: Resets all devices on the peripheral bus

Peripherals of the SharcFIN include the SDRAM controller, the ADSP-21160 DSPs, and any devices attached to the peripheral bus. The reset that is passed to the ADSP-21160s (via the **ResetO** signal) is conditioned according to Analog Devices' recommendation in the ADSP-21160 anomaly list dated 05/25/2001 to properly initialize the ADSP-21160s' core clock PLL. This anomaly list requires guaranteeing that the ADSP-21160 reset is at least 60 microseconds long and disabling the clock for 10 microseconds during that time. The clock is disabled by the clock enable line (**ClkEn**). If the reset signal coming to the conditioner is less than 60 microseconds, the conditioner keeps the ADSP-21160s in reset for 60 microseconds. If the reset signal is longer than 60 microseconds, the conditioner keeps the ADSP-21160s in reset until the reset signal is cleared. Figure 2-6 illustrates the SharcFIN's reset circuitry.

Figure 2-6
SharcFIN Reset Circuitry



Performing a Total Reset (TR)

The first PCI side reset source is the PCI reset line **RST**, which resets every function in the chip and all peripherals attached to it. It is issued by the PCI bus to reset all PCI devices on the bus. The SharcFIN is completely reset when **RST** from the PCI bus is active. All registers are initialized to the default conditions, and all FIFOs are emptied. After **RST** is deasserted, the chip will come out of reset after four PCI clocks and will then respond to PCI accesses.

The PCI Bus reset input **(RST)**, when active, causes all PCI bus outputs to be asynchronously tri-stated (or floated), which is a requirement of the *PCI Local Bus Specification* (Revision 2.2). All internal state machines are held in idle states until **RST** is deasserted.

To maintain hot-plug compatibility, all reset internal functions will not come out of the reset state until an idle phase is detected on the PCI bus. An idle phase is defined as **FRAME** inactive and **IRDY** inactive. Because the SharcFIN is a nonvolatile device, no power-up configuration delays are necessary.

RST is deglitched and synchronized to the PCI clock, **CLK**, prior to its connection to the PCI core. **RST** must be active for a minimum number of cycles of **CLK** and **LClk** (see Section 3.9.4). During any resetting function, both the PCI clock, **CLK**, and the user clock, **LClk**, must be running. This requirement may have implications for subsystems that have PLLs on the clocks.

Performing a Host Total Reset (HTR)

The second PCI side reset source is called a Host Total Reset. The PCI host may initiate a Host Total Reset by writing a "1" to the control register address offset 0x3A, bit[0]. This is *not* a complete chip reset. In addition to resetting the SHARC interface of the chip, a Host Total Reset resets the state of the control registers from 0x00 – 0x3F, including the PCI side DMA engines. It will reset the entire chip with the exception of PCI configuration space; therefore all of the hot plug/BIOS configuration will still be intact.

The PCI core will automatically clear the write of "1." After "1" has been written, the chip's control will go into reset until both clock domains have gotten into reset, and it will respond to PCI access after a given number of clocks. A simple way to recognize that the SharcFIN has come out of reset is to read the vendor device ID register in PCI configuration space until the correct value is read back.

Resetting the SHARC Interface and All Peripherals

Resetting via the PCI Host. The PCI host may initiate a SHARC interface reset by writing a "1" to the PCI control register address offset 0x3A, bit 32. The SHARC interface of the chip will be held in reset, but the PCI core will still respond to PCI accesses. This action will reset the SHARC interface and all peripherals, including the ADSP-21160 DSPs. The reset is cleared by writing a "0" to 0x3A bit [32].

Resetting via an External Reset Source. The SharcFIN's external reset line (**WD_Rst**) is typically attached to a watchdog chip and includes resets from the watchdog or an external reset pin. This signal resets the entire SHARC interface and all peripherals.

Resetting Peripherals Attached to the SharcFIN

Resetting the SDRAM Controller. Bit[3] of the ***SD Size Configuration*** register resets the SDRAM controller in the SharcFIN. The reset that is passed to the SDRAM controller is conditioned to last a maximum of two **LClk** clock cycles to allow the SDRAM controller to continue to refresh the SDRAM so its contents are not lost. It is typically not necessary to reset the SDRAM controller. For details on the ***SD Size Configuration*** register, refer to the section entitled "SDRAM Size Configuration Register" on page 185.

Resetting Devices on the Peripheral Bus. Bit 5 of the ***Peripheral Bus Configuration*** register resets all devices on the peripheral bus, including the flash memory, UART, and the peripheral bus reset line. For details on the ***Peripheral Bus Configuration*** register, refer to the section entitled "Peripheral Bus Configuration Register" on page 181.

2.6.2 Watchdog Interface

The SharcFIN is designed to be connected to a Dallas Semiconductor 1832 watchdog chip. The chip has three inputs and two reset outputs. The active low reset output is intended to connect to the SharcFIN's **$\overline{\text{WD\_Rst}}$** pin (see Table 4-2 on page 4-200 for signal description). One of the inputs can be connected to an external push-button reset. The other two inputs (TCL and TDLY) are intended to be connected to the SharcFIN's **WD_TCL** and **WD_TDLY** pins respectively. These two inputs allow the SharcFIN to implement a watchdog reset function using ADSP-21160 flags to tickle the watchdog. The ***Watchdog Configuration*** register (0x43) controls how these two lines are used.

Bits [1:0] of the ***Watchdog Configuration*** register enable the watchdog function and set the required tickle frequency. Bits [7:4] enable 21160-1 F1, 21160-2 F1, 21160-3 F1, or 21160-4 F1 (respectively) as sources for the tickle function. The section entitled "Watchdog Configuration Register" on page 182 explains the register in more detail. Refer to the DS1832 data sheet for details on the watchdog's operation.

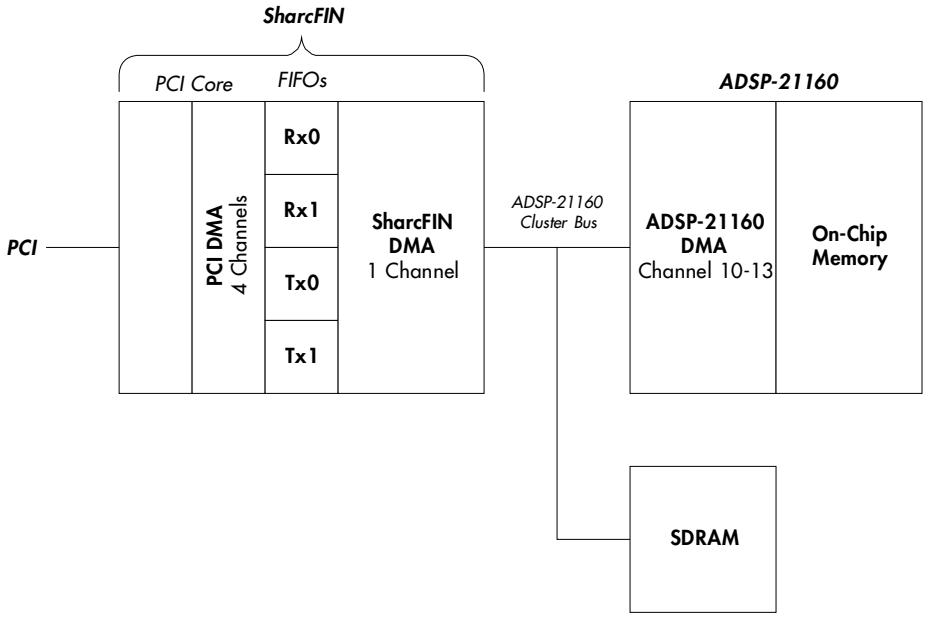
2.7 Bus Mastering and DMAs

The bus mastering capabilities of the SharcFIN consist of two parts. The first involves moving data between the DMA FIFOs and the PCI bus. The other involves moving data between the ADSP-21160 cluster bus and the DMA FIFOs. This section covers the following topics:

- Bus mastering (section 2.7.1)
- Single PCI accesses (section 2.7.2)
- DMAs between the DMA FIFOs and the PCI bus (section 2.7.3)
- DMAs between the ADSP-21160 cluster bus and the DMA FIFOs (section 2.7.4)

Figure 2-7 illustrates the SharcFIN's DMA channels.

Figure 2-7
DMA Channels in the SharcFIN ASIC



2.7.1 Bus Mastering

The presence of the SharcFIN allows the Bittware DSP board to master the PCI bus. This enables the DSP board to have direct access to any PCI device in the system, allowing it to read or write data without involving the host computer. You may even opt to ignore a provider's OS driver for the PCI device, which becomes important in applications in which real-time performance is required since the host computer's operating system will often schedule the servicing of PCI devices indeterminately. Moreover, PCI devices without drivers for the chosen OS can be controlled by creating a host interface to the device via the DSP board.

Note *In the discussion to follow, SharcFIN control and status register addresses are given as byte offsets from the base (MS2 from the processor's perspective; BAR0 from the PCI bus), but the registers indicated are 64 bits long.*

The SharcFIN contains five independent bus mastering channels:

- Two for data moving from the ADSP-21160 cluster bus to PCI
- Two for data moving from PCI to the ADSP-21160 cluster bus
- One channel for moving single bytes, dwords, or quadwords of data to/from PCI (SPCI)

The five channels are labeled as follows:

1. Receive Channel 0
2. Receive Channel 1
3. Transmit Channel 0
4. Transmit Channel 1
5. Single PCI Access (SPCI)

Each channel, with the exception of SPCI, has its own 64 quadwords deep FIFO, and all five channels can perform full 64-bit addressing and 64-bit data transfers. When chain mode is enabled, Receive Channel 0 and Transmit Channel 0 are dedicated to chaining.

The host processor or local processor sets up a DMA transfer by setting the PCI address and transfer count. The function is enabled when a "1" is written to the DMA Start/Status bit of the respective channel (control registers address 0x24). The SharcFIN will request access to the PCI bus, and when access is obtained, it will burst the data starting at the location specified in the Master Write Address register.

The SharcFIN will generate a Dual Address Cycle (DAC) if the PCI address is beyond the lower 4GB range (upper 32 bit not equal to zero), and it will not generate a DAC if the address is within the lower 4GB range (according to the PCI specification). This is automatically determined by the PCI core.

2.7.2 Single PCI Accesses

The SharcFIN supports a single PCI access channel for performing single PCI reads and writes. To perform PCI reads and writes, tell the SharcFIN which address to read or write, provide the data (for a write), and then request the transfer. On a read completion, the data is available in a buffer to be read. As with the DMAs, the SharcFIN is designed for 64-bit transfers and alignment. You can make it perform any number of byte width transfers by specifying which of the 8 bytes of the 64-bit access are to be enabled. However, you will need to align the data in the 64-bit word and use the 64-bit aligned address.

Note *Single PCI accesses can only be performed from the ADSP-21160s. Unlike DMAs, single PCI accesses can not be performed by the host.*

Single PCI Access Mastering

The SharcFIN has a mastering channel that allows a single quadword of data to/from the SharcFIN using any of the commands shown in Table 2-6. This mastering mode is intended for use by a local processor to perform any or all of the following tasks that require the movement of small amounts of data:

- Type 0 or 1 configuration cycles
- Chain descriptor table initialization
- Random memory read/writes
- Random I/O read/writes
- Special cycles
- Interrupt acknowledge

This master can be programmed to use the following commands:

- Memory Read Multiple (MRM)
- Memory Read Line (MRL)
- Memory Write and Invalidate (MWI)

Table 2-6
Single PCI Mastering Commands

C/BE#[3:0]	Command Type	SharcFIN Capability
0000	Interrupt Acknowledge	Master
0001	Special Cycle	Master
0010	I/O Read	Master/Target
0011	I/O Write	Master/Target
0100	Reserved	N/A
0101	Reserved	N/A
0110	Memory Read	Master/Target
0111	Memory Write	Master/Target
1000	Reserved	N/A
1001	Reserved	N/A
1010	Configuration Read	Master/Target
1011	Configuration Write	Master/Target
1100	Memory Read Multiple	Master/Target
1101	Dual Address Cycle	Master/Target
1110	Memory Read Line	Master/Target
1111	Memory Write and Invalidate	Master/Target

These commands, however, imply that multiple pieces of data will be transferred. Since only a single quadword at a time can be transferred using the SPCI controller, it doesn't make any sense to use these commands with this function. The SPCI master always has the highest priority of all mastering channels, regardless of the arbitration selected in the ***Arbitration Mode*** register.

The following steps are needed to execute a single PCI Access:

For a write (SHARC interface to PCI):

1. Write the target address to the ***Single PCI Address*** register[63:0] (0x26).
2. Write the target data to the ***Single PCI Data*** register[63:0] (0x28).
3. Write byte enables, command, and start to ***Single PCI Access***, ***Control*** (0x08, bits [61:48]).
4. Wait until the ***start*** bit (0x20, bit[61]) is low.

For a read (PCI to SHARC interface):

1. Write the target address to ***Single PCI Address*** register[63:0] (0x26).
2. Write byte enables, command and start to ***Single PCI Access, Control*** (0x08, bits[61:48]).
3. Wait until the ***start*** bit (0x08, bit [61]) is low.
4. Data can then be read from ***Single PCI Data*** register[63:0] (0x28).

Bit[2] in the address register is ignored, and bit[2] in the resulted PCI transaction is generated from the byte-enables automatically. Address bits 1:0 are passed directly to the PCI bus, and they should always be "00" when doing a memory or configuration transaction. For I/O transactions, they should be equal to the location of the first active byte, indicated by the byte enables, within the doubleword. The following are some examples:

- Memory transactions:

  Address register [1:0] = "00", byte enables = "11000000"; this will generate a 32-bit transaction and PCI address [2:0] will be "100".

  Address register [1:0] = "00", byte enables = "11111111"; this will generate a 64-bit transaction and PCI address [2:0] will be "000".

- I/O transactions:

  Address register [1:0] = "10", byte enables = "11000000"; this will generate a 32-bit transaction, and PCI address [2:0] will be "110".

  Address register [1:0] = "00", byte enables = "11111111"; this will generate two 32-bit transactions, and PCI address [2:0] will be "000" and "100" respectively.

  Address register [1:0] = "01", byte enables = "00000010"; this will generate a 32-bit transaction, and PCI address [2:0] will be "001".

When setting up the SPCI transaction, the data must be aligned into the correct byte lanes in the quadword that is reflected by the byte enables. This action is required even for configuration cycles.

Single PCI Accesses from an ADSP-21160 DSP Board

From the viewpoint of the Bittware DSP board, SPCI accesses are as described above, but with some important considerations. The DSP board's cluster bus, which is shared by all processors, SDRAM, and the SharcFIN, is 32-bit addressable. This means that the addresses to the SharcFIN registers, indicated above as byte-oriented offsets, must be divided by four (4) to reorient them to 32 bits. Furthermore, each 64-bit register is seen as a pair of 32-bit register halves.

The SharcFIN is mapped to the ADSP-21160's MS2, which is the base address. On the ADSP-2116x processors, MS2 must be calculated according to the MSIZE field in the ADSP-21160's SYSCON register. See the *ADSP-21160 SHARC DSP Hardware Reference* (Analog Devices) for more information on calculating the address of MS2. For this discussion, we can assume the base address of the SharcFIN is known and is indicated by *ms2_base*. The examples that follow will use a C Language-like syntax for clarity.

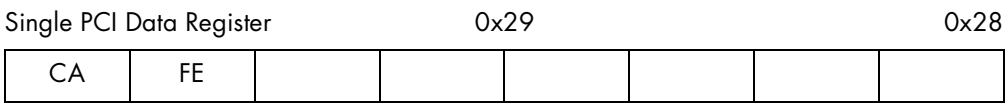
Example 1: Performing an SPCI 16-bit Write Access. The following example shows how to perform a SPCI 16-bit write access from an ADSP-21160 processor, relating the steps to those stated above. Suppose your PCI target address, *pci_addr,* is 0xF5000006, and your data to write, *data_val,* is 0xCAFE.

1. Write target address to the ***Single PCI Address*** register[63:0] (0x98). The ***Single PCI Address*** register is at 32-bit offset 0x26, which is the lower half of the 64-bit register. Although the target address is 32 bits long, clear the upper 32 bits in the address register. Align the address to 64 bits by clearing the lower 3 bits, then write the aligned address with two 32-bit accesses:

   ```
   *(ms2_base + 0x26) = pci_addr & 0xFFFFFFF8;

   *(ms2_base + 0x27) = 0;
   ```

2. Write target data to the ***Single PCI Data*** register[63:0] (0xA0). The ***Single PCI Data*** register is at 32-bit offset 0x28, which is the lower half of the 64-bit register. Place your data in the correct location within the 64 bits of the data register. Since your target address is 0xF5000006, write the 16 bits of data to the upper two byte positions in the data register as shown. The SPCI mastering channel can be used to read or write any number and any combination of bytes in a quadword.



   You can accomplish this with one 32-bit access to the upper half of the register:

   ```
   *(ms2_base + 0x29) = data_val << 16;
   ```

3. Write byte enables, command, and start to the ***Single PCI Access, Control*** register (0x20, bits [61:48]). The ***Single PCI Access and Control*** register is at 32-bit offset 0x08; however, since this example will write bits in the upper half, point to offset 0x09. Set the upper two byte enables (0x00C00000), a 32-bit write command (0x17000000), and set the start bit (0x20000000) with the value 0x37C00000.

   ```
   *(ms2_base + 0x9) = 0x37C00000;
   ```

Note *The preceding command could have been a 64-bit write (0x07000000), but in the case of a transfer that is known to be 32 bits or less, a few cycles may be saved by specifying the 32-bit write.*

4. Wait until the *start* bit (0x08, bit[61]) is low.

   Poll the ***Single PCI Access and Control*** register until the start bit clears:

```
while(*(ms2_base + 0x09) & 0x20000000)
{
}
```

Example 2: Performing an SPCI 8-bit Read Access. This second example shows how to perform a SPCI 8-bit read access from an ADSP-21160 processor, relating the steps to those stated above. Suppose your PCI target address, *pci_addr,* is 0xF8000003, and you wish to read your byte of data into *read_val*.

1. Write the target address to the ***Single PCI Address*** register[63:0] (0x26). This step is identical to that of the write example above.

```
*(ms2_base + 0x26) = pci_addr & 0xFFFFFFF8;

*(ms2_base + 0x27) = 0;
```

2. Write byte enables, command, and start to the ***Single PCI Access, Control*** (0x20, bits [61:48]). The ***Single PCI Access and Control*** register is at 32-bit offset 0x08; however, for this example you will need to write bits in the upper half, so point to offset 0x09. Set the *byte 3 enable* bit (0x00080000), a 32-bit read command (0x16000000), and set the *start* bit (0x20000000) with the value 0x36080000.

```
*(ms2_base + 0x9) = 0x36080000;
```

Note *The above command could have been a 64-bit read (0x06000000), but in the case of a transfer that is known to be 32 bits or less, a few cycles may be saved by specifying the 32-bit read.*

3. Wait until the *start* bit (0x08, bit [61]) is low. Poll the ***Single PCI Access and Control*** register until the start bit clears:

```
while(*(ms2_base + 0x09) & 0x20000000)
{
}
```

4. Data can then be read from the ***Single PCI Data*** register[63:0] (0x28). The ***Single PCI Data*** register is at 32-bit offset 0x28, which is the lower half of the 64-bit register. Since your data will appear in byte 3, the lower data register is where you'll find it.

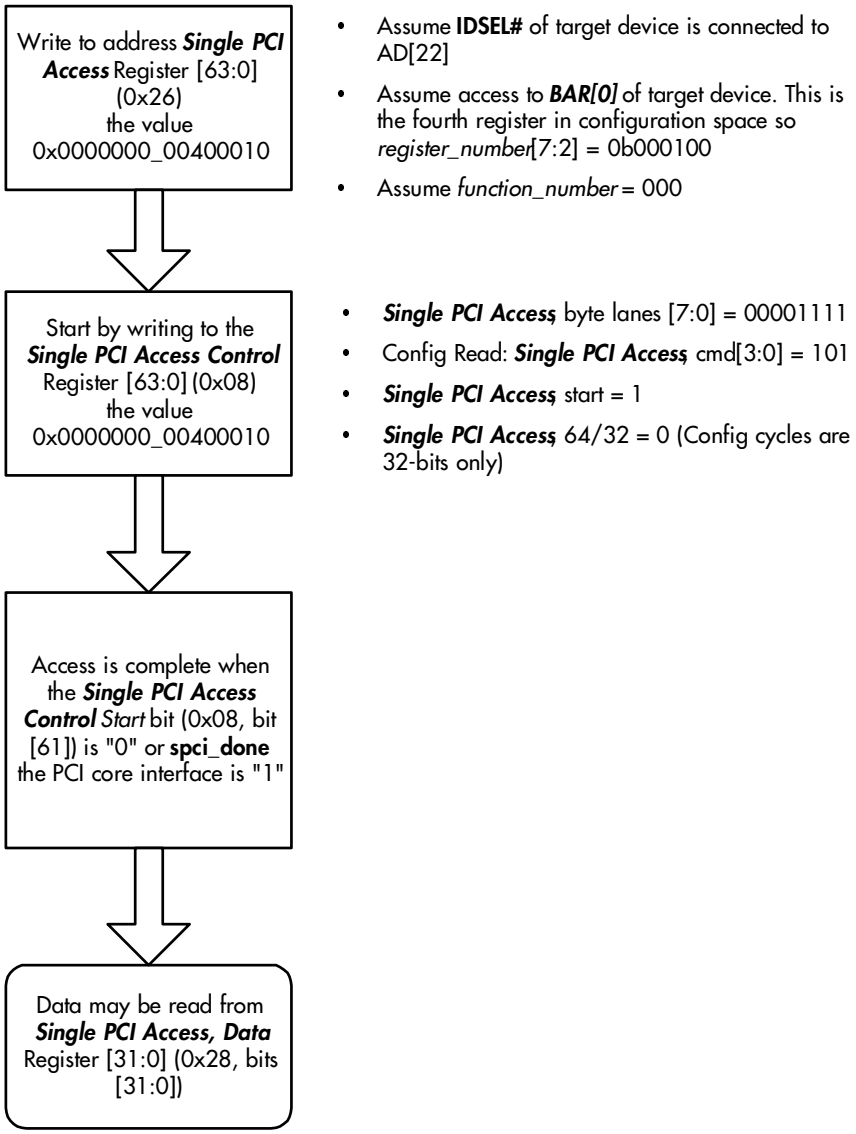
```
read_val = (*(ms2_base + 0x28) >> 24) & 0xFF;
```

Note *On all SPCI mastering operations using the SharcFIN, if you do not know whether the destination of the transaction is 32 or 64 bits, assume 64 bits, and the SharcFIN will automatically determine the width using* **REQ64** *and* **ACK64**. *If you do know the bus width of the destination, you can potentially save a few clock cycles by giving the SharcFIN this information by setting (32-bit) or clearing (64-bit) control bit [60] of register 0x20.*

Generating Type 0 or Type 1 Configuration Cycles

Under the control of an ADSP-21160 processor, the SharcFIN can generate either Type 0 or Type 1 configuration cycles. This allows the ADSP-21160 to configure an embedded system and act as the host processor. Figure 2-8 on page 2–48 shows the steps necessary to complete a Type 0 configuration cycle. Type 0 and Type 1 configuration accesses are limited to 32-bit.

Figure 2-8
Steps Necessary to Perform a Type 0 Configuration Cycle



2.7.3 PCI Side DMAs

PCI side DMAs involve moving data between the DMA FIFOs and the PCI bus. The PCI side of the SharcFIN supports a 5-channel DMA engine. See Figure 2-7 for an illustration of SharcFIN's DMAs.

Transmit DMA Channels 0/1

"Transmit" means moving data from the DMA FIFOs to PCI. The SharcFIN has two transmit channels. Each channel has a dedicated FIFO that is 72 bits wide by 64 deep. The FIFOs are named Transmit FIFO0 and Transmit FIFO1. Figure 2-9 on page 2–50 and Figure 2-10 on page 2–51 show the steps necessary to start a transmit DMA operation. In summary,

1. A host or local processor writes the PCI address at which the operation will start to the ***Master Write Address*** register.
2. Next, a transfer count (in bytes) is written to the ***Master Write Count*** register. The operation is started when a "1" is written to the *DMA Start/Done, xmt0/1* bit in control register 0x24.
3. Data may be placed in a Transmit FIFO at any time, and multiple transactions may be stacked. A related transmit channel does not need to be enabled before data is placed in its FIFO. Unwanted or unused data may be flushed. (See "Flushing the Transmit FIFOs" on page 51.)
4. Unaligned transfers are not supported. The registers ***Master Write Address 0***[63:0] and ***Master Write Address 1***[63:0] must be 64-bit word aligned addresses; and the registers ***Master Write Transfer Count 0***[31:0] and ***Master Write Transfer Count 1***[31:0] must be multiples of 64-bit word transfers expressed in bytes.

Figure 2-9
Steps Necessary to Start a Transmit0 DMA Operation (Control Register Offset 0x00)

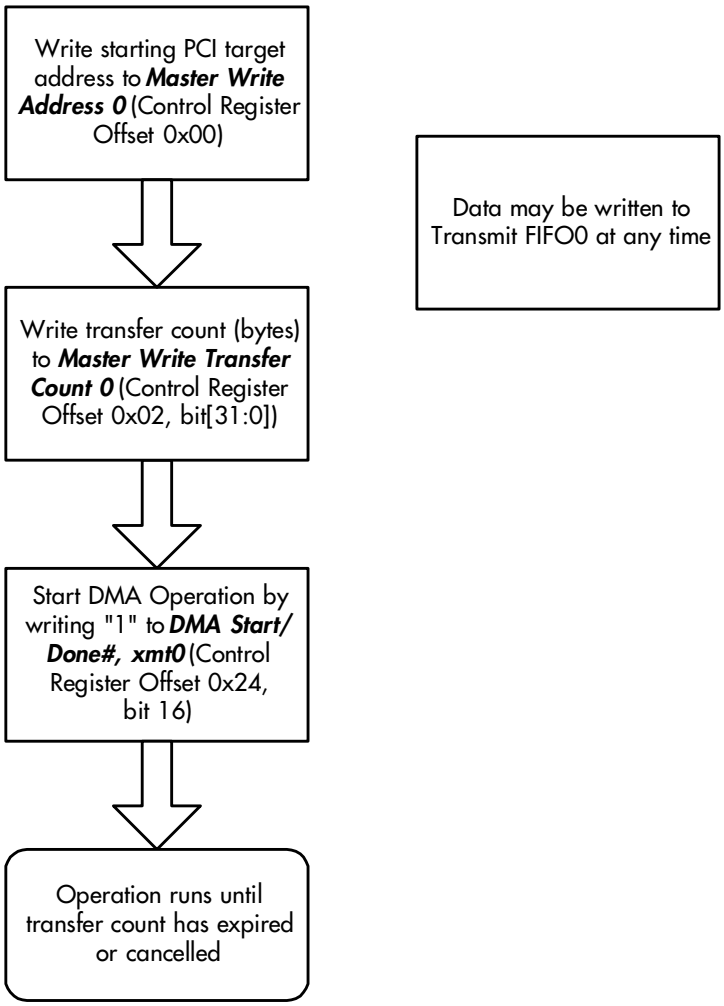
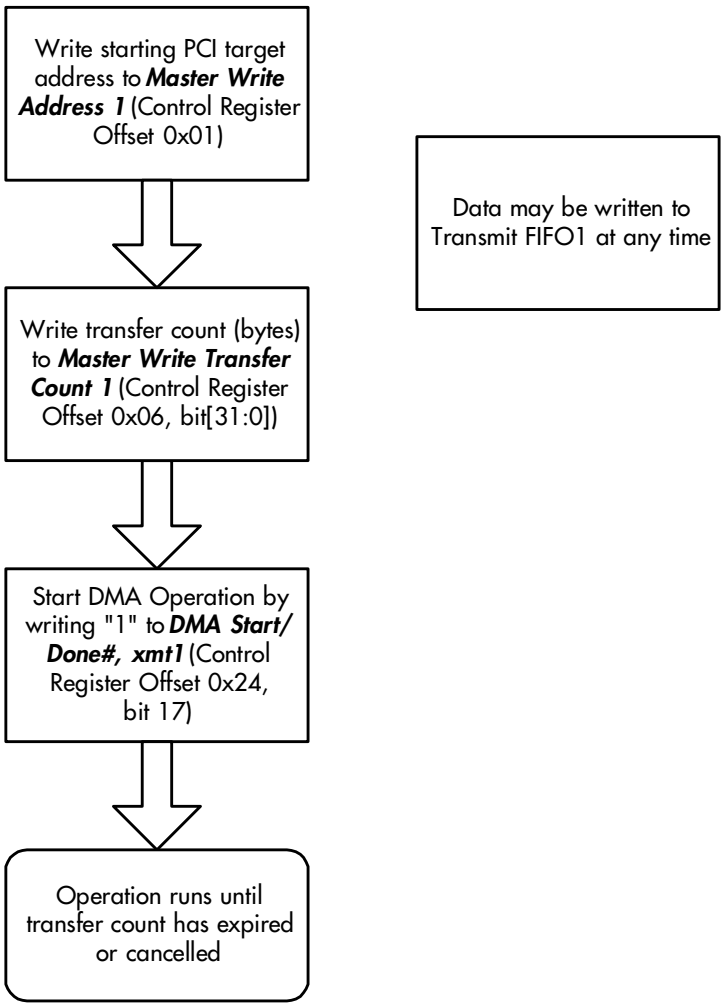


Figure 2-10
Steps Necessary to Start a Transmit1 DMA Operation (Control Register Offset 0x10)



Flushing the Transmit FIFOs. Unwanted or unused data may be flushed from Transmit FIFO0 or Transmit FIFO1 by writing a "1" to the correct bit in the control register from the host processor located on PCI or from a local processor. The flush bits are located at offset 0x34. *FIFO Flush xmt0* is bit[21] and *FIFO Flush xmt1* is bit[22].

The process of flushing is not immediate; it is dependent on the amount of data in the FIFO. A flush is complete after the flush bit, when read, is inactive. The maximum possible time it could take to flush a FIFO is sixty-five **CLK** cycles**,** plus two **LClk** (ADSP-21160 cluster bus clock) cycles after assertion of the FIFO flush bit.

If the flush operation is initiated from PCI, it could take an additional four **CLK** (PCI clock) cycles plus four **LClk** (ADSP-21160 clock) cycles plus any additional time needed to flush posted writes from the Target Write Post FIFO. Both FIFOs may be flushed at the same time. A Transmit DMA Channel must be idle to execute a flush on its FIFO.

Monitoring Status of a Transmit DMA. You can monitor the status of a Transmit DMA operation two ways:

- Read the *DMA Start/Done, xmt0* bit (0x24, bit[16]) for Transmit Channel 0.
- Read the *DMA Start/Done, xmt1* bit (0x24, bit[17]) for Transmit Channel 1.

The bits will be active as long as the operation is running. You can also read the *Master Write Count Status* bits. These 32-bit registers indicate how many bytes are left to transfer. *Master Write Count Status 0[31:0]* is at control register 0x02, bit[63:32] and *Master Write Count Status 1[31:0]* is at control register 0x06, bit[63:32].

Canceling/Stopping a Transmit DMA. You can halt a Transmit DMA operation before all data has been transferred or before the requested transfer count has reached zero. Unfortunately, this action is tricky since stopping a Transmit DMA operation depends partially on the target that is receiving the data. The SharcFIN provides a bit for each transmit channel that allows you to halt execution. Two styles of halting are possible: Graceful and PUNT@#$!.

Figure 2-11 and Figure 2-12 on page 2–53 are flowcharts diagramming the Graceful method. The Graceful method requires you to stop writing to the Transmit FIFO. Once writing to the Transmit FIFO has stopped, wait until all transactions have completed by monitoring the *Pipeline Not Empty* bit in control register 0x20. The Graceful method is the preferred method of halting a transmit DMA.

Figure 2-13 and Figure 2-14 on page 2–54 describe the PUNT@#$! method. If you use the PUNT@#$! method, the last pending transfer may not complete if the target had issued a retry and a DMA Cancel was asserted before the Transmit controller re-issued the command. This, technically, is a violation of the PCI Specification since all attempted transactions are explicitly required to complete, and this violation could result in a temporary or permanent hang condition.

Figure 2-11
Graceful Method Flowchart for Actions Necessary for Canceling/Stopping Transmit DMA Channel 0

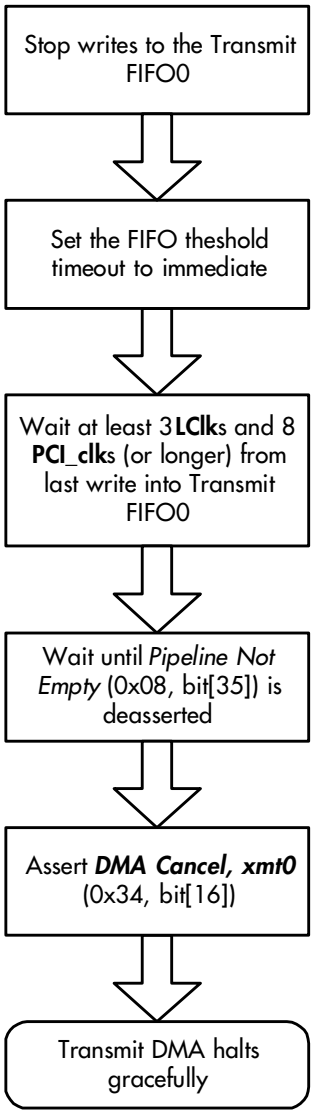


Figure 2-12
Graceful Method Flowchart for Actions Necessary for Canceling/Stopping Transmit DMA Channel 1

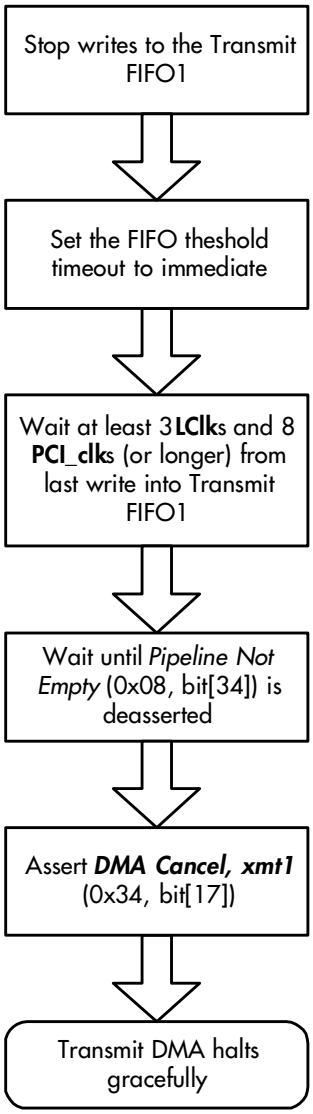


Figure 2-13
PUNT@#$! Method Flowchart for Actions Necessary for Canceling/ Stopping Transmit DMA Channel 0

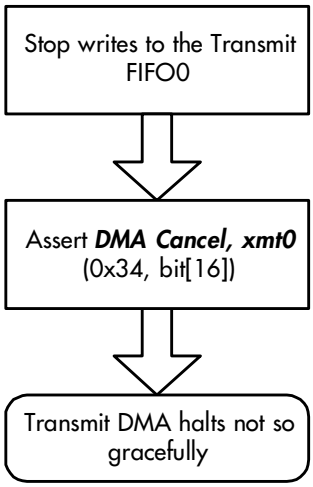
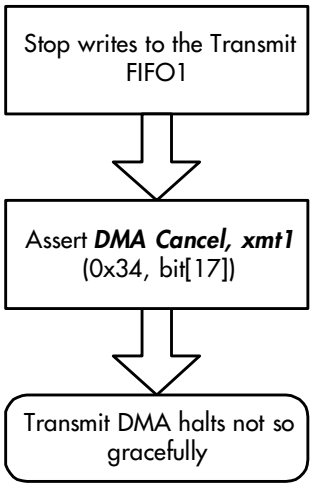


Figure 2-14
PUNT@#$! Method Flowchart for Actions Necessary for Canceling/ Stopping Transmit DMA Channel 1



Bursting Control. The SharcFIN has two programmable parameters that control the flow of data on the PCI bus for mastering operations. The first parameter is an almost empty flag that triggers a transmit DMA controller to request a PCI transaction. The second is a timeout that overrides this trigger if too much time has expired.

Transmit FIFO Almost Empty Flags. Each transmit FIFO has a programmable empty flag that can be set to trigger when the FIFO contains 0–63 quadwords of data. A Transmit DMA controller will wait to request access to the PCI bus until the related programmable empty flag indicates that a sufficient number of quadwords of data are available in the FIFO to burst. This guarantees that each PCI transaction initiated by a transmit DMA controller will burst at least the number of quadwords set in the ***Almost Empty Flag*** register.

When the number of quadwords that remain in the transfer count less than the value programmed in this register, the transmit controller will ignore this trigger and transfer the data to PCI as it is loaded into the FIFO.

Note *The two transmit FIFOs have a three-quadword buffer after the FIFO. When the small buffer is not full, data is immediately pulled out of the FIFO and placed into that buffer. The programmable full flag is only relative to the amount of data in the FIFO, not the buffer. The value used for the programmable pointer should be three less than the desired burst length, in quadwords.*

The register for Transmit FIFO0 is at control address 0x1A, bits[37:32] and Transmit FIFO1 is at control register address 0x1A bits[45:40].

FIFO Timeout. The FIFO time-out is a global programmable variable that affects all four mastering FIFOs. This variable overrides the almost empty flags, in the case of the Transmit FIFOs, or the almost full flags, in the case of the Receive FIFOs, if too much time has expired. The four possible values are charted in Table 2-7.

Table 2-7
FIFO Threshold Timeout

	Time@ 33 MHz	Time @ 66 MHz
00	0 µs	0 µs
01	30.7 µs	15.4 µs
10	1.966 ms	0.983 ms
11	∞	∞

Each Transmit channel has a separate timer; if this timer expires, the related transmit controller will request access to the PCI bus and will transfer all data that is in the FIFO to the target until the FIFO is empty or the target drives **$\overline{\text{STOP}}$**. The timer is re-started when either the FIFO is empty or **$\overline{\text{TRDY}}$** is driven by the destination device on PCI.

PCI Error. If an error occurs during the PCI transaction, the controller will record the error in the control registers. Please refer to the section entitled “PCI Error” on page 67.

Receive DMA Channels 0/1

“Receive” means moving data from the PCI bus to the DMA FIFOs. The SharcFIN has two receive channels, each with a dedicated FIFO. The FIFOs are named Receive FIFO0 (74 bits wide by 64 deep) and Receive FIFO1 (72 bits wide by 64 deep). Figure 2-15 and Figure 2-16 on page 2–56 show the steps necessary to start a Receive DMA operation. In summary:

1. A host or local processor writes the PCI address where the operation will start to the ***Master Read Address*** register.
2. Next, a transfer count (in bytes) is written to the ***Master Read Count*** register.
3. The operation is started when a “1” is written to the *DMA Start/Done, rcv0/1* bit in control register 0x24.
4. Unaligned transfers are not supported on either channel. The registers ***Master Read Address 0***[63:0] and ***Master Read Address 1***[63:0] must be 64-bit word aligned addresses; and the registers ***Master Read Transfer Count 0[31:0]*** and ***Master Read Transfer Count 1***[31:0] must be multiples of 64-bit word transfers in bytes.

Figure 2-15
Steps Necessary to Start Receive Channel 0 Mastering Operation

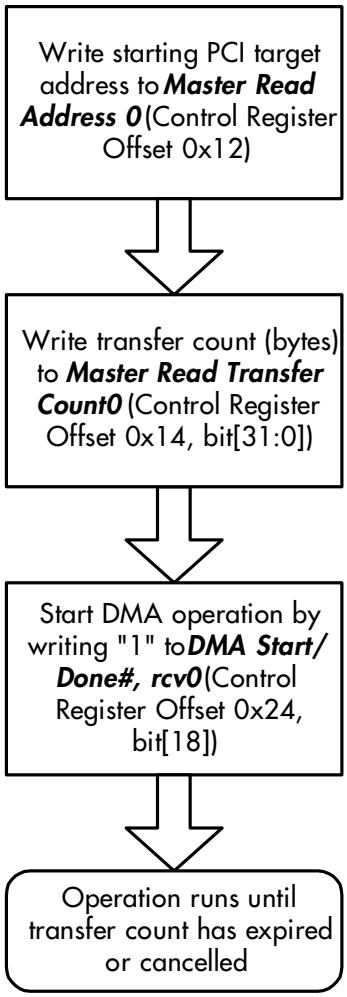
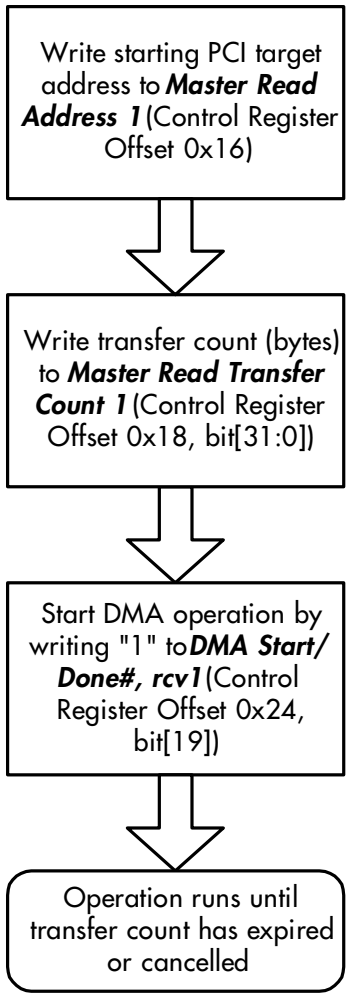


Figure 2-16
Steps Necessary to Start Receive Channel 1 Mastering Operation



Flushing the Receive FIFOs. Unwanted or unused data may be flushed from Receive FIFO0 or Receive FIFO1 by reading the data out of the FIFO until the empty flag is asserted. The empty flags can be monitored at control register 0x02, bits 17 and 19 for FIFO0 and 1 respectively.

Monitoring Status. Status of a Receive DMA operation may be monitored two ways. Either the *DMA Start/Done, rcv0* bit (0x24, bit[18]) may be read for Receive Channel 0, or the *DMA Start/Done, rcv1* bit (0x24, bit[19]) may be read for Receive Channel 1. The bits will be active as long as the operation is running.

The *Master Read Count Status* bits can also be read. These 32-bit registers indicate how many bytes are left to transfer. *Master Read Count Status 0[31:0]* is at control register 0x14, bit[63:32] and *Master Read Count Status 1[31:0]* is at control register 0x18, bit[63:32].

Canceling/Stopping Receive DMA. A receive DMA operation may be halted before all data has been transferred (before the requested transfer count has reached zero). Unfortunately this is tricky, since stopping a receive DMA operation is partially dependent on the target that is providing the data. The SharcFIN provides a bit for each receive channel that allows you to halt execution. Two styles of halting are possible:

- Graceful
- PUNT@#$!

Figure 2-17and Figure 2-18 are flowcharts diagramming the Graceful method. The Graceful method requires the user to stop reading from the receive FIFO so that the FIFO fills. Once the FIFO is full, you can assert the *DMA Cancel, rcv0/1* bit in control register 0x08. The Graceful method is the preferred method of halting a receive DMA.

Figure 2-19 and Figure 2-20 describe the PUNT@#$! method. If the PUNT@#$! method is used, the last pending transfer may not complete if the target had issued a retry and a DMA cancel was asserted before the receive controller re-issued the command. This, technically, is a violation of the PCI specification and could have serious repercussions since all attempted transactions are explicitly required to complete. It could lead to a temporary or permanent hang condition on PCI.

Bursting Control. The SharcFIN has two programmable parameters that allow you to control the flow of data on the PCI bus. The first parameter is an almost full flag that triggers a Receive DMA controller to request a PCI transaction. The second is a time-out that overrides this trigger if too much time has expired.

Receive FIFO Almost Full Flags. Each Receive FIFO has a programmable full flag that can be set to trigger when the FIFO contains 0–63 empty locations. A Receive DMA controller will wait to request access to the PCI bus until the related programmable full flag indicates that a sufficient number of quadwords of data are available to be written to the FIFO. This guarantees that each PCI transaction initiated by a Receive DMA controller will burst at least the number of quadwords set in the ***Almost Full Flag*** register.

Figure 2-17
Graceful Method Flowchart for Actions Necessary for Canceling/Stopping Receive DMA Channel 0

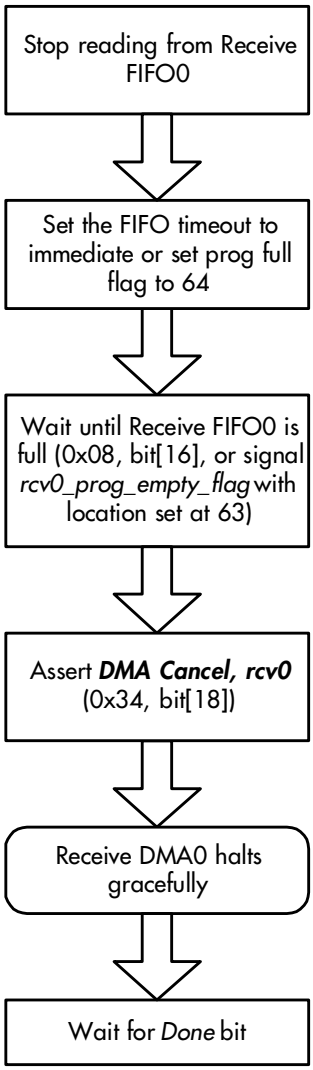


Figure 2-18
Graceful Method Flowchart for Actions Necessary for Canceling/Stopping Receive DMA Channel 1

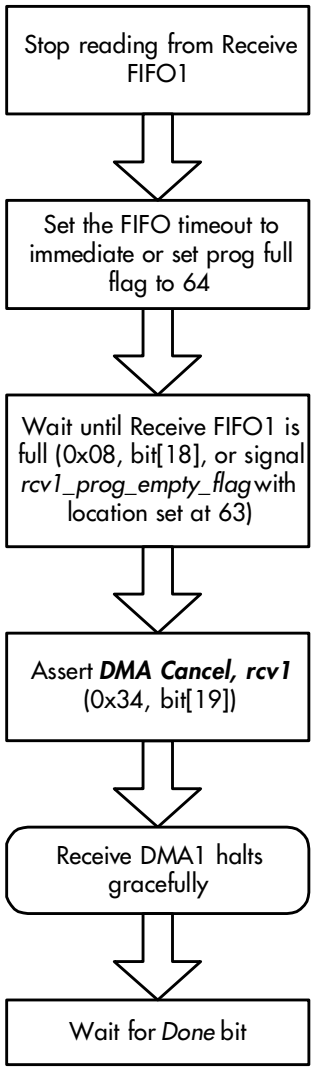


Figure 2-19
PUNT@#$! Method Flowchart for Actions Necessary for Canceling/ Stopping Receive DMA Channel 0

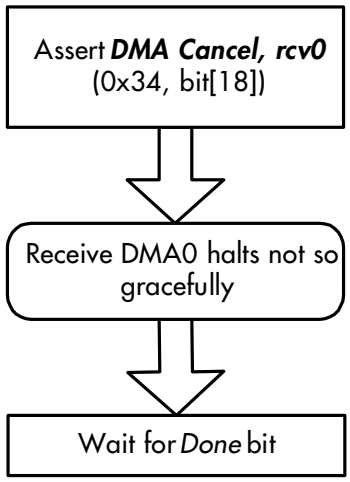
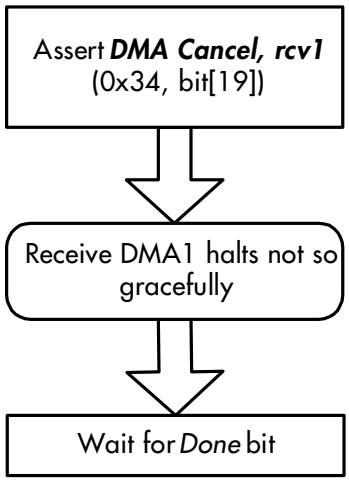


Figure 2-20
PUNT@#$! Method Flowchart for Actions Necessary for Canceling/ Stopping Receive DMA Channel 1



When the number of quadwords that remain to be transferred is less than the value programmed in this register, the receive controller will ignore this trigger and transfer the data from PCI and load it into the FIFO. The register for Receive FIFO0 is at control address 0x1A, bits[21:16] and Receive FIFO1 is at control register address 0x1A, bits[29:24].

PCI Error. If an error occurs during the PCI transaction, the controller will record the error in the control registers. Please refer to the section entitled "PCI Error" on page 67.

Chaining Mode DMA

The SharcFIN has a very powerful chain mode function. When chain mode is active, Receive FIFO0 and Transmit FIFO0 are dedicated to chaining operations. In chaining mode, the host or local processor sets up descriptor blocks in PCI memory that are composed of four quadwords of information. Figure 2-22 on page 2–61 shows the details of the content of the chain descriptor blocks.

All descriptor information is passed through Receive FIFO0 and can be latched in the SHARC interface or read by a local processor. The presence of descriptor data at the output of Receive FIFO0 is determined by reading the fifth byte lane at address offset 0x08 of the control registers. A non-zero value on the tag bits indicates the presence of descriptor data.

The address of the first chain register is placed by the host or local processor in the "Master Read Address 0" at control register address offset 0x12. This address must be on a quadword aligned address (bits[2:0] must be 0). Chain operation is initiated by writing a "1" to the *chain start* bit at bit[20] of register 0x24. As long as the chain operation is running, bit[20] of register 0x24 will be "1." The next one bit over, bit[21], can be read to determine if the last chain descriptor has been read and the end-of-chain tag has been set. Figure 2-22 illustrates

the chain mode. Once the end-of-chain tag has been observed, the chain descriptors can be deleted or written over.

Chain mode is also sometimes described as "scatter/gather" (see Figure 2-21). The entire second quadword is intended to be used as a local start address pointer but can be used to pass any other information that is interesting to the ADSP-21160 hardware.

Note *If a chain descriptor has a pointer address that points to a previous descriptor block, this mode can be made to loop indefinitely. Other vendors refer to this variation on chaining as "shuttle mode DMA."*

Figure 2-21
Chain Mode Flow Chart

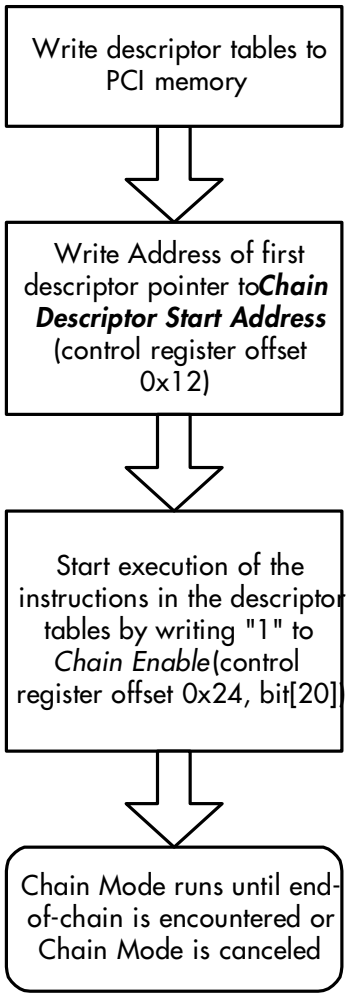
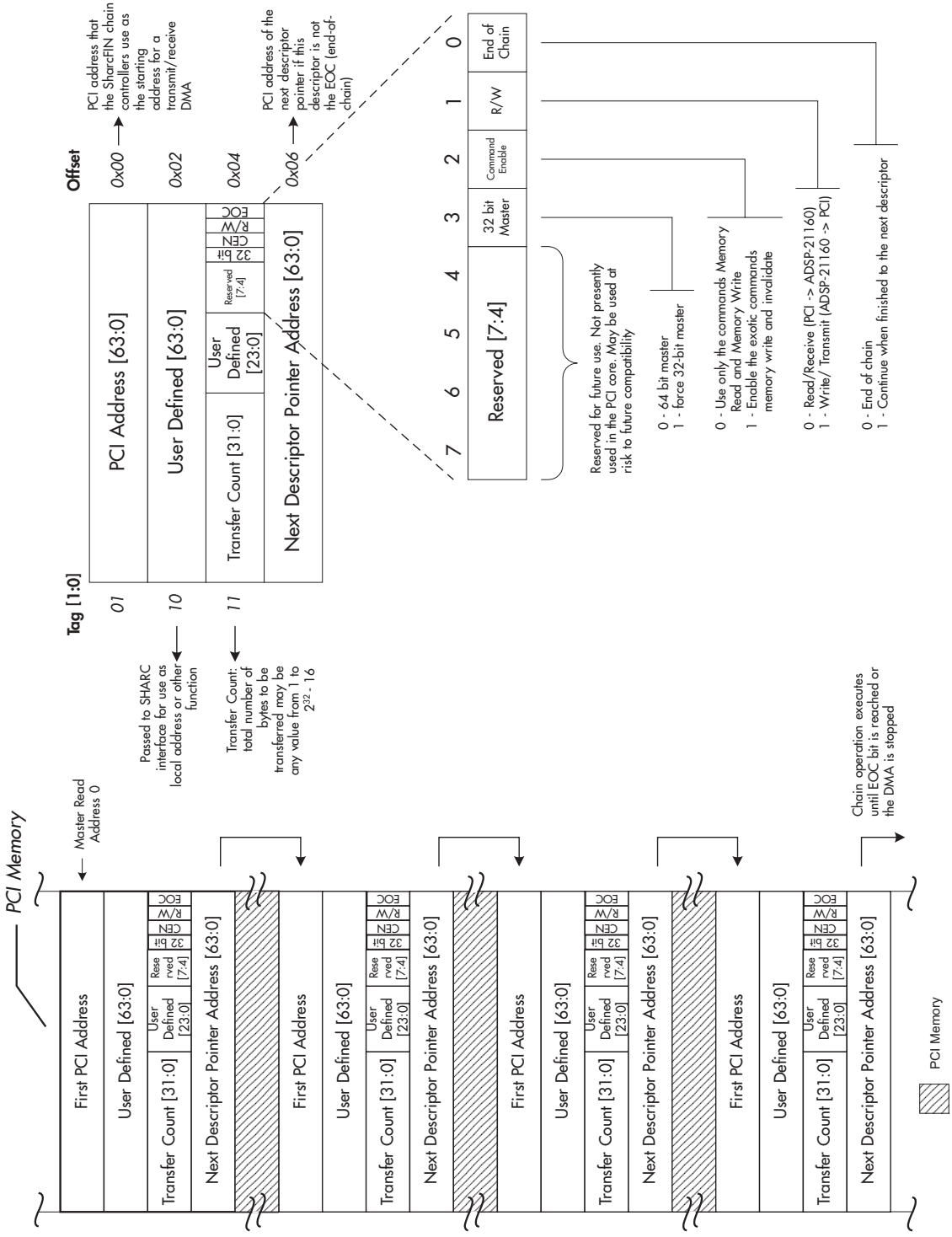


Figure 2-22
Chain Mode Description



Twenty-four bits of the third quadword are also user-defined and can be used for any purpose you see fit. A unique descriptor ID that can be read by the local or host processor, for example, would be a good way to monitor the status of the chaining operation. The interrupt controller can be set to interrupt the PCI processor, the local processor, or both.

Chain mode DMA transfers must be 64-bit word aligned like normal DMAs. A maximum of 2^{32}–16 bytes (0xFFFFFFF0) of information may be transferred with each descriptor.

PCI Error. If an error occurs during the PCI transaction, the controller will record the error in the control registers. Please refer to the section entitled "PCI Error" on page 67.

2.7.4 ADSP-21160 Cluster Bus DMAs

SHARC interface DMAs involve moving data between the ADSP-21160 cluster bus and the DMA FIFOs. The SharcFIN supports two primary methods of moving data between the cluster bus and the FIFOs. The first method involves using a single-channel DMA engine built into the SharcFIN. The second method involves using an ADSP-21160 DSP to move the data via either core operations or an external DMA channel.

The SharcFIN DMA is highly optimized for speed. It has been shown to achieve the following sustained rates:

Table 2-8
SharcFIN DMA Sustained Data Rates

	SharcFIN DMA	ADSP-21160 DMA
Tx	260 Mbytes/sec	160 Mbytes/sec
Rx	140 Mbytes/sec	140 Mbytes/sec

Note *These rates were achieved on a 64-bit, 66 MHz system using the most optimal DMA condition. Performance will vary based on DMA parameters and the capabilities of the device being communicated with on the PCI bus.*

Using the SharcFIN-Based DMA Engine

To use the SharcFIN's DMA engine to perform DMAs between the ADSP-21160 cluster bus and the DMA FIFOs, you will need to set up two registers: the ***DMA Address*** register and the ***DMA Configuration*** register. Refer to "DMA Address Register" on page 186 and "DMA Configuration Register" on page 186 for an overview of all the bits in these registers.

Setting the DMA Address Register. The first register to set up is the ***DMA Address*** register, which is located at 32-bit word offset 0x48 (byte offset 0x120) from BAR0. Bits 0, 28, 29, 30, and 31 of this register are unwritable and fixed at 0, and all accesses are 64-bit aligned operations. The ***DMA Address*** register sets the address of the first 64-bit word to be

transferred by the DMA engine. It also determines whether the data transferred will be from SDRAM or one of the ADSP-21160s.

Addresses at 0x800000 and above are considered to be SDRAM accesses. The SDRAM window register does not apply to DMA addresses because the DMA engine has full access to all of the SDRAM.

The memory layout from this register appears the same as the MMS memory view of the ADSP-21160s. Therefore, ADSP-21160-1 memory appears at 0x140000 to 0x15FFFF, and ADSP-21160-2 appears at 0x240000 to 0x25FFFF, etc. Broadcast write space is also valid at 0x740000 to 0x75FFFF.

This register is incremented as the DMA progresses, allowing you to monitor the DMA engine's current address. You must reset this register each time if you wish to repeat a DMA on the same address range as the previous transfer. This register cannot be written to while the DMA start bit is set, but it can be read from at any time.

Setting the DMA Configuration Register. The ***DMA Configuration*** register is located at 32-bit word offset 0x49 (byte offset 0x124) from BAR0. Bits in this register set various parameters of the transfer (direction, transfer count, etc.) and start and stop the transfer.

Table 2-9
Contents of the DMA Configuration Register

Bit	Name	Description
15:0	DMACntB[15:0]	DMA transfer count (in 64-bit words) bits. All are settable.
22:16	DMA Stride B[6:0]	Stride or address increment bits. These bits will typically be set to 0x01.
23	Unused	Unwritable; fixed at 0.
24	DMAStart	Setting the bit to 1 starts the DMA; it resets to 0 when the DMA is complete.
25	DMA Channel Select	Selects the ESP channel being operated on (0 or 1)
26	DMA Direction	Selects whether an ESP transmit or receive DMA is being performed. 0 = ESP receive DMA (PCI to 21160/SDRAM) 1 = ESP transmit DMA (21160/SDRAM to PCI)
27	DMA Interrupt	If this bit is set at the start of the DMA, The SharcFIN generates an interrupt in the interrupt multiplexer on completion of the DMA.
28	Burst Disable	1 = Disable bursting on the ADSP-21160 side. Must be set to 1 when the address increment > 0x01. If it is set when the address increment <= 0x01, it functions but will slow things down. 0 = Bursting enabled
29	DMA Buslock	1 = SharcFIN requests the ADSP-21160 cluster bus when the start bit is set, and once it obtains the bus, keeps it until the DMA completes.
31:30	DMA Xfer Length B[1:0]	Set the DMA transfer length. These bits must be set along with the bits in the PCI control register at 0x1A and 0x1B (0x68 byte address). See details below.

The minimum size for a DMA that goes from PCI to the ADSP-21160 are eight 64-bit words. The minimum is one 64-bit word going the other way. Set the DMA count bits to 0x0000 to transfer the maximum of 0x10000 (65,536) 64-bit words.

The increment is somewhat confusing because it is a 64-bit word increment. In other words, if you set it to 0x01, the address register (32-bit address) actually increments by 2 on each transfer. The sequence of locations transferred if the increment register is set to 0x01 and the address register is set to 0x800000 is as follows:

Xfer #Addresses Xfered
1 0x800000, 0x800001
2 0x800002, 0x800003
3 0x800004, 0x800005
ETC

The sequence of locations transferred if the increment register is 0x02 and the address register is set to 0x800000 is as follows:

Xfer #Addresses Xfered
1 0x800000, 0x800001
2 0x800004, 0x800005
3 0x800008, 0x800009
ETC

While the start bit is set to 1, the ***DMA Address*** register (0x48) is read only, and *only* the start bit may be written to in the ***DMA Configuration*** register (0x49). Setting the start bit to 0 allows you to clear a DMA in progress. The start bit can be polled to monitor completion of the DMA, but keep in mind that the poll rate can significantly affect transfer rates. Interrupts are better but can have more latency. The sequence for starting a DMA should be the following:

1. Set up and start PCI DMA.
2. Set the ***DMA Address*** register.
3. Set the ***DMA Configuration*** register.

The *start* bit can be set at the same time as all the other bits in the ***DMA Configuration*** register.

The DMA interrupt can be mapped into an interrupt by setting bit 14 in the appropriate interrupt multiplexer register. *Any* write to the ***DMA Configuration*** register once the interrupt is set (i.e. DMA is done) clears the DMA interrupt.

DMAs with the bus lock bit (*DMA Buslock*) set the result in the highest transfer rate but prevent the ADSP-21160s from accessing their bus until the DMA completes. The host can still access the SharcFIN and the ADSP-21160 bus normally while the DMA has the bus locked.

Bits [31:30] control how much data the DMA engine transfers at a time. The four possible settings correspond to:

1. Eight 64-bit words
2. Sixteen 64-bit words
3. Thirty-two 64-bit words
4. Sixty-four 64-bit words

The different values affect two things. First is overall throughput. A setting of 2 (thirty-two 64-bit words) is optimal for speed, then 3, 1, and 0 is slowest (your mileage may vary depending on PCI bus size, speed and the abilities of what you are DMAing to/from on the PCI side). This setting also affects bus use. Transferring data in chunks of eight 64-bit words

will use the bus more frequently, but for shorter periods of time than if you transfer sixty-four 64-bit words at a time.

A PCI control register must also be set with a value corresponding to the following setting (X = don't care):

Table 2-10
Settings for PCI Control Registers at Offsets 0x1A and 0x1B

DMA to use PCI Receive channel 0

B31	B30	Value that must be written to register at location 0x1A (0x68)
0	0	0xXXXXXX07
0	1	0xXXXXXX0F
1	0	0xXXXXXX1F
1	1	0xXXXXXX3F

DMA to use PCI Receive channel 1

B31	B30	Value that must be written to register at location 0x1A (0x68)
0	0	0xXXXX07XX
0	1	0xXXXX0FXX
1	0	0xXXXX1FXX
1	1	0xXXXX3FXX

DMA to use PCI Transmit channel 0

B31	B30	Value that must be written to register at location 0x1B (0x6C)
0	0	0xXX07XXXX
0	1	0xXX0FXXXX
1	0	0xXX1FXXXX
1	1	0xXX3FXXXX

DMA to use PCI Transmit channel 1

B31	B30	Value that must be written to register at location 0x1B (0x6C)
0	0	0x07XXXXXX
0	1	0x0FXXXXXX
1	0	0x1FXXXXXX
1	1	0x3FXXXXXX

Using the ADSP-21160-Based DMA Engine

The ADSP-21160s can be used to provide data to the DMA FIFOs using either core-based or DMA-based transfers. Transfers from the ADSP-21160s to the DMA FIFOs must use 64-bit non-bursting mode. The transfer must also be set up with an external increment of 0 and the address to MS2 base + 0x2C, 0x2E, 0x30, or 0x32 depending on the channel being used (RX0, RX1, TX0, TX1 respectively).

Using the ADSP-21160s to provide data to/from the DMA FIFOs presents a potential bus lock condition. If the transmit FIFOs are full or the receive FIFOs are empty, and an ADSP-21160 attempts to read/write to them, the SharcFIN holds the read or write transaction off by lowering the **H_Ack** (ADSP-21160 Ack) line. This transaction cannot be aborted; therefore, the ADSP-21160 cluster bus is inaccessible until space clears up in the FIFOs (or is filled by the receive FIFOs). If the ADSP-21160 bus is unavailable in this situation, when a target transaction requires the ADSP-21160 bus, a deadlock condition exists that cannot be resolved. A system's reaction to this condition varies with the system's BIOS but typically results in a hung computer. Potential workarounds to this problem include making sure the host does not attempt to access the board while a DMA is in progress. If this cannot be avoided, another solution is to break the ADSP-21160 DMA into sixty-four 64-bit transfers, making sure that the transmit FIFO is empty (or the receive FIFO is full) before each transfer is initiated. This guarantees that the ADSP-21160 bus never becomes unavailable. Examples of this workaround are available from BittWare. The SharcFIN-based DMA engine does not have this limitation.

When using the ADSP-21160 DMA engine, always start the PCI side DMA before the ADSP-21160 DMA. Starting the PCI side DMA first will keep the DMA from filling the transmit FIFO or reading an empty receive FIFO, making the bus unavailable for the core to write to the SharcFIN to initiate the PCI side DMA.

2.7.5 Bus Mastering Issues

PCI Error

The following is a list of situations in which the DMA engine will not complete its transfer successfully.

- Data Parity Error
- Master Abort
- Target Abort
- Retry Timeout

If any of the above occurs, the DMA engine will automatically be stopped, the corresponding error bits (address 0x25) will be set for the condition, and the DMA status bit will be set back to "done".

A Retry Timeout error happens when the controller has retried a transaction for a given number of times before giving up and signaling an error through the control registers. The number of retries that the device will attempt before canceling the transaction is programmed by two bits in control register 0x34 called *Max Retry[1:0]*. The four values are, in PCI transactions:

- 0x100 transactions => approximately 30us given an idle bus
- 0x40000 transactions => ~30ms
- 0x10000000 transactions => ~30s
- Infinite => will never cancel the transaction

The suggested value of *Max Retry* is the ~30ms value for debugging a system, but use the "infinite" value for production systems in which the latency might be very large. When you are using either the ~30s or the infinite value, a condition may occur in which the system decides that the transaction will never complete, and the transaction must be canceled. Four *PCI Error Status* bits (0x24, bits[59:56]) show which type of error occurred in the PCI transaction. They are cleared when a '1' is written to the bit.

Each DMA channel (including chaining and SPCI) has an *Error Status* bit (0x24, bits [53:48]), which is set when a PCI error is detected during that channel's DMA transaction. These bits are read-only. They are updated when the corresponding channel is used again.

When a PCI error occurs, the data that was in the FIFO will not be flushed. Therefore, for the retry and target abort cases, you may restart the transaction with the next address and a decremented transfer count (read the transfer count status for the number of bytes already transferred).

Canceling Bus Mastering Operations

The SharcFIN has five separate DMA engines. Four of them are associated with FIFOs (XMT0/1, RCV0/1). The last one has only a single quadword of data that it can transfer per transaction (SPCI).

The four DMA engines that have FIFOs associated with them have a cancel bit (control register address 0x34) that is useful when the system decides to cut a transaction short or when it must overcome an infinite retry problem. The SPCI DMA engine does not have the cancel bit because all of its data (or buffer space) is already available in the chip, and it attempts to transfer its data immediately and will not stop until it has been transferred. The only way to cancel the SPCI transaction is to change the ***Max Retry*** register value to "~30us" and let it execute the 256 transactions and fail quickly.

Note *This same method does not necessarily cancel a transaction that is associated with a FIFO. It would only cancel it if it was stuck in infinite retries. If one of the four DMA engines is waiting on one of the internal FIFOs to have data/space available, no transactions will be generated on the bus, and there will therefore be no retries to count. In that case, the system should use one of the cancel bits to end the DMA operation.*

Master Arbitration Control

Five possible masters can request access to the PCI bus from the SharcFIN:

1. Transmit 0
2. Transmit 1
3. Receive 0
4. Receive 1
5. SPCI

The SharcFIN supports two separate arbitration priority schemes: Round Robin and Fixed. The SPCI engine always has priority regardless of the scheme used. The PCI host or local processor select the arbitration mode by writing one of the following values to the ***Arbitration Mode Select*** registers at offset 0x34 bits[49:48]:

Arbitration Mode[1:0]

00 ROUND_ROBIN
01 SCHEME_PRIORITIZED
10 Reserved
11 Reserved

When a master obtains access to the bus, the master will keep the bus until one of the following conditions occurs:

1. Transaction completes without a retry
2. No more data to transmit in transmit FIFO *or* FIFO full in receive FIFO
3. Transfer count==0
4. Master abort
5. Target abort
6. Parity error
7. Master operation cancelled by user
8. Latency counter expired

Round Robin. In the Round Robin scheme, the direct master always has the highest priority, then the arbiter will cycle through the four masters in the following order:

1. Transmit 0
2. Transmit 1
3. Receive 0
4. Receive 1

Fixed Priority. In the Fixed Priority scheme, each channel has a fixed priority. A larger, 2-bit value programmed into the *DMA_arbitration_priority[1:0]* (address offset 0x34) means higher priority. Priorities that are set to be equal are arbitrated in the following order:

1. Transmit 0 highest priority
2. Transmit 1 middle-high
3. Receive 0 middle-low priority
4. Receive 1 lowest priority

Master Latency Counter Enable/Disable

The SharcFIN contains a master latency counter that controls the amount of time that a master may have the PCI bus. When the timer has expired and **GNT** is deasserted, the master must relinquish control of the bus.

The purpose of this timer is to prevent a single master from hogging the bus, preventing other devices from getting any bandwidth. The PCI Specification limits this counter to eight bits, which in turn limits the size of a potential burst.

The SharcFIN has a control register bit that allows its masters to ignore the latency counter. Ignoring the latency counter is a direct violation of the PCI Specification but allows for coarser control over PCI arbitration issues for embedded systems. With the latency counter disabled, a single master can maintain control over the bus indefinitely. The bit is called *LATENCY_EN* and is at control register address 0x34.

Bus Mastering in a 32-Bit Environment

The SharcFIN can operate in a 32-bit or 64-bit PCI environment. See section 2.6 for a detailed description of how the SharcFIN determines the bus width at reset time. The SharcFIN can dynamically interface to both 32-bit and 64-bit PCI devices on a transaction-by-transaction basis.

If any of the five masters of the SharcFIN are initiating data transactions to a target device that is known to be 32-bits, providing this information to the SharcFIN can result in some performance gains. If the width of the target device is unknown, program the SharcFIN to assume that the target device is 64-bits, and the SharcFIN will determine the proper bus width dynamically on each transaction using the PCI signals **ACK64** and **REQ64**.

The following control registers are applicable:

Single PCI Access (SPCI)
QLPCI_(U/P)_64_SPCI_ADDR (0x08, bit[59])
1 Assume destination is 32 bits
0 Automatically determine bus width using **REQ64**. Initially assume 64 bits.

Receive DMA Channel 1
QLPCI_P_64_RCV_1_FORCE_32_ADDR (0x08, bit[35])
1 Automatically determine destination bus width using **REQ64**. Initially assume 64 bits.
0 Assume destination is 32 bits.

Receive DMA Channel 0
QLPCI_P_64_RCV_0_FORCE_32_ADDR (0x08, bit[34])
1 Automatically determine destination bus width using **REQ64**. Initially assume 64 bits.
0 Assume destination is 32 bits.

Transmit DMA Channel 1

QLPCI_P_64_XMT_1_FORCE_32_ADDR (0x08, bit[33])

1 Automatically determine destination bus width using **REQ64**. Initially assume 64 bits.

0 Assume destination is 32 bits.

Transmit DMA Channel 0

QLPCI_P_64_XMT_0_FORCE_32_ADDR (0x08, bit[32])

1 Automatically determine destination bus width using **REQ64**. Initially assume 64 bits.

0 Assume destination is 32 bits.

The chain descriptor contains a bit to control this function also. See Figure 2-22 on page 2–61.

2.8 I^2C Interface and Serial Controller

The SharcFIN's I^2C/Serial controller integrates some of the most common peripheral requirements right into the SharcFIN. Uses include data communications, SharcFIN interconnection, hardware configuration and identification, and user non-volatile storage.

The SharcFIN contains two I^2C interfaces. One connects to onboard configuration EEPROMs that allow software to determine the board configuration information, and provide a small amount of non-volatile memory storage. The register at 0x46 controls this I^2C bus. The other is connected to the PMC+ site and is intended as a simple command and control I/O interface for PMC+ peripherals. The register at 0x47 controls it.

2.8.1 How the I^2C Interfaces Function

An I^2C interface consists of two signals: a clock and a data line. This interface was standardized by Philips Electronics, and detailed information about it can be found on their website (www.philips.com). I^2C capable peripherals also often contain excellent information about specifically accessing that device via I^2C.

The SharcFIN directly connects to the clock and data line, and bits in each interface's control register allow the SharcFIN to drive either or both of these lines low or to leave them tri-stated. Both the clock and data lines are externally pulled up, allowing the buses to operate as both bi-directional and multi-master as described in the Philips Electronics specification.

The I^2C control registers are 4 bits wide. Bit [0] controls the clock line of the bus. Writing a 0 to this bit enables the SharcFIN driver and drives this line to 0. Writing a 1 to this register disables the SharcFIN driver and allows this line to return to a 1 by its pull-up resistor if no other I^2C peripherals are driving it low. Reading bit [0] returns the actual state of the clock line on the I^2C bus. Bit [1] operates similarly to bit [0] except that it relates to the data line. Bit [2] is read only; when read, it returns the state of the SharcFIN driver on the clock line (1 = disable or tri-stated, 0 = enabled and driven low). Bit [3] operates the same as bit [2] except that it relates to the data line. This arrangement allows you to determine on a single read both the actual state of the bus, and the state of the SharcFIN's drivers to easily determine if the SharcFIN, another peripheral, or nothing is driving the I^2C bus.

2.8.2 Accessing Serial EEPROMs

On BittWare Hammerhead boards, the first I^2C bus (0x46) is connected to two serial EEPROMs that contain configuration about the board. These serial EEPROMs are the Board Configuration EEPROM on the board and the Serial Presence Detect (SPD) EEPROM on the SDRAM module.

Table 2-11
Information for I^2C Protocol

Serial EEPROM	I^2C Device Number
Board Configuration	2
SPD EEPROM	0

SPD EEPROM on SDRAM Module

The SPD EEPROM is a read-only EEPROM module found on most SO-DIMM memory modules. This EEPROM contains standard configuration information about the SO-DIMM. For a full description of the contents of the EEPROM, refer to Appendix E of the SPD section of the *JEDEC JESD21-C standard* (www.jedec.org). Information on a specific module's SPD EEPROM can be found on the module manufacturers web site. The contents used to set SDRAM registers in the SharcFIN are:

Table 2-12
Contents Used to Set SDRAM Registers

Name	Offset (in bytes)
Number of Rows	5
Refresh Rate/Type	12
Bank Density	31

Your total module size (in MB) can be found by: *Number of Rows* × *Bank Density* × *4*

After issuing a board reset, the HIL sets the SharcFIN's ***SD Size Config*** register (offset 0x45) according to the tables below.

Table 2-13
How the HIL Writes the SD Size Config Register (0x45)

Number of Rows, Bank Density	SDRAM Size Config Register bits 1 and 0 (offset 0x45)
1, X	00
> 1, 0x10 (64 MB)	01
> 1, 0x20 (128 MB)	10
> 1, 0x40 (256 MB)	11

Refresh Rate/Type	SDRAM Size Config Register bit 2 (offset 0x45)
0x01	1
0x02	Unsupported SO-DIMM
0x81	1
0x82	Unsupported SO-DIMM
All other values	0

Board Configuration EEPROM

The Board Configuration EEPROM is compatible with Microchip Technology Inc.'s 24LC01B/02B. Refer to their documentation for specifics on accessing this type of EEPROM. The lower 128 bytes of the EEPROM are read/write, and the upper 128 bytes are read-only. In the upper 128 bytes, the information in Table 2-14 is stored.

Table 2-14
Board Configuration EEPROM Information

Offset	Name	Data Type	Description
0x80	struct_level	ULONG	Structure revision level
0x84	board_type	USHORT	HIL board type
0x86	dsp_type	USHORT	HIL DSP type of processors
0x88	dsp_rev	USHORT	DSP revision
0x8A	num_dsps	USHORT	Number of DSPs on board
0x8C	wait_reg	ULONG	WAIT register value
0x90	msize	ULONG	MSIZE bits value (for SYSCON)
0x94	memory_code[4]	USHORT	Memory code for MS spaces
0x9C	device_num	ULONG	Unused
0xA0	serial_num	ULONG	Board serial number
0xA4	factory_date	ULONG	Date this EEPROM was burned
0xA8	lot_number	ULONG	Production lot number of board
0xAC	board_cfg_level[2]	USHORT	Board configuration level = board_cfg_level[0].board_cfg_level[1]
0xB0	sdram_spd[2]	USHORT	SDRAM SPD setup: sdram_spd[0] = number of rows, sdram_spd[1] = bank density
0xB4	part_number[16]	UCHAR	Part number string (should end with '\0')
0xC4	last_written_date	ULONG	Date this EEPROM was last written
0xC8	num_writes	ULONG	Number of full EEPROM writes
0xCC	last_written_by[4]	UCHAR	Initials of last writer (should end with '\0')
0xD0–0xF4	spare0 - spare9	ULONG	Unused
0xF8	valid_mask	ULONG	Shows which fields are valid
0xFC	check_um	ULONG	EEPROM checksum of valid fields

Chapter 3
Register Descriptions

The SharcFIN has two sets of registers. One set, the PCI configuration registers, configures the PCI interface. The other set, the chip control registers, configures both the PCI and SHARC interfaces. The PCI configuration registers are only accessible by PCI configuration access. The chip control registers are broken into two groups: the PCI control registers and the SHARC interface control registers; both groups are accessible by PCI (BAR0) and the ADSP-21160 cluster bus (MS2).

3.1 SharcFIN Memory Map

3.1.1 Overview

The SharcFIN maps into PCI and the ADSP-21160 cluster bus. It uses PCI Base Address Registers (BARs) to map its various parts onto the PCI bus. BAR0 maps the SharcFIN's control registers, BAR1 maps to the peripheral bus (Flash and dual UART), BAR2 maps to the ADSP-21160's MMS space, BAR3 is unused in a 32-bit environment and used for the upper 32 bits of address in a 64-bit addressable system, and BAR4 maps to the SDRAM.

The SharcFIN maps into the ADSP-21160 cluster bus space using the MS (memory select) lines of the ADSP-21160s. MS0 maps to the SDRAM, MS1 maps to Flash, dual UART, and peripheral bus, and MS2 maps to the SharcFIN control registers.

This section provides an overview of the PCI and ADSP-21160 memory mapping of the SharcFIN. All addresses are shown as offsets from the appropriate BAR or MS. Table 3-1 gives an overview of how the SharcFIN maps to the PCI and ADSP-21160 cluster buses.

Table 3-1
Overview of How the SharcFIN Maps to PCI and ADSP-21160 Buses

PCI Base Address Register	Description	21160 Memory Select	Description
BAR0	SharcFIN control registers	MS0	SDRAM
BAR1	Peripheral bus (Flash, UART)	MS1	Peripheral bus (Flash, UART)
BAR2	ADSP-21160 MMS	MS2	SharcFIN control registers
BAR4	SDRAM		

Even though the following sections list both 32-bit (word) and byte addresses, some BARs should be accessed in specific ways from the PCI side. Table 3-2 shows how to access those BARs.

Table 3-2
Accessing BAR0–BAR4 From the PCI Side

BAR	Access	Description
BAR0	Read/Write	Byte or 32-bit word accesses for all registers
BAR1	Read/Write	Byte accesses for all registers*
BAR2	Read Only	Byte or word accesses
	Write Only	Word accesses only†
BAR4	Read Only	Byte or word accesses
	Write Only	Word accesses only†

* Word accesses will produce erroneous data.
† Byte writes will corrupt the rest of the word.

3.1.2 Accessing System Settings and Configuration Registers

BAR0 = MS2 = System settings and configuration registers

BAR0 from the PCI interface and MS2 from the ADSP-21160 DSPs map to system settings and configuration registers in the SharcFIN. Table 3-3 gives the PCI and ADSP-21160 offset addresses for accessing system settings and configuration registers.

Table 3-3
PCI and ADSP-21160 Addresses for System Settings and Configuration Registers

PCI 32-bit Offset From BAR0	PCI Byte Offset From BAR0	ADSP-21160 Offset From MS2	Description
0x00 – 0x5F	0x000 – 0x17F	0x00 – 0x5F	Chip control registers (PCI and ADSP-21160)

3.1.3 Accessing the Flash, UART, and Peripheral Bus

BAR1 = MS1 = Flash/UART/Peripheral bus

BAR1 from the PCI interface maps to the Flash, dual UART, and peripheral bus. MS1 from the ADSP-21160 cluster bus also maps to the Flash, dual UART, and peripheral bus.
Table 3-4 gives the PCI and ADSP-21160 offset addresses for accessing the Flash, UART, and peripheral bus.

Note *BAR1* ***must*** *be accessed a byte at a time from the PCI side. Word accesses will produce erroneous data.*

Table 3-4
PCI and ADSP-21160 Addresses for Flash, UART, and Peripheral Bus

PCI Byte Offset From BAR2	ADSP-21160 Offset From MS1	Description
0x000000 – 0x1FFFFF	0x000000 – 0x1FFFFF	Flash
0x200000 – 0x20000F	0x200000 – 0x20000F	UART
0x200010 – 0x2FFFFF	0x200010 – 0x3FFFFF	Reserved
0x300000 – 0x3FFFFF	0x400000 – 0x5FFFFF	Peripheral Bus

3.1.4 Accessing Multiprocessor Memory Space

BAR 2 = MMS = flat map of Multiprocessor Memory Space

BAR2 from the PCI and MMS from the ADSP-21160 cluster bus allow access to the ADSP-21160 multiprocessor memory space. Table 3-5 gives the PCI and ADSP-21160 offset addresses for accessing the MMS.

Table 3-5
PCI and ADSP-21160 Addresses for Multiprocessor Memory Space

PCI 32-bit Offset From BAR2	PCI Byte Offset From BAR2	ADSP-21160 Address	Description
0x00000 – 0x0FFFFF	0x0000000 – 0x03FFFFF	0x000000 – 0x0FFFFF	Reserved
0x10000 – 0x1FFFFF	0x0400000 – 0x07FFFFF	0x100000 – 0x1FFFFF	21160-1 MMS space (ADSP-21160 ID 1)
0x20000 – 0x2FFFFF	0x0800000 – 0x0BFFFFF	0x200000 – 0x2FFFFF	21160-2 MMS space (ADSP-21160 ID 2)
0x30000 – 0x3FFFFF	0x0C00000 – 0x0FFFFFF	0x300000 – 0x3FFFFF	21160-3 MMS space (ADSP-21160 ID 3)
0x40000 – 0x4FFFFF	0x1000000 – 0x13FFFFF	0x400000 – 0x4FFFFF	21160-4 MMS space (ADSP-21160 ID 4)
0x50000 – 0x7FFFFF	0x1000000 – 0x1FFFFFF	0x500000 – 0x7FFFFF	Reserved

3.1.5 Accessing SDRAM

BAR 4 = MS0 = SDRAM

You can see a window of 16 Mbytes of SDRAM from the PCI bus. The ***SD Window*** register allows you to select which 16 MB window is currently visible. The register is located at word/ADSP-21160 offset 0x4A in BAR0/MS2. Table 3-6 gives the PCI and ADSP-21160 offset addresses for accessing a 64 MB bank of SDRAM, and Table 3-7 gives addresses for a 128 MB bank.

Note *The addresses listed in Table 3-6 and Table 3-7 only apply to the given 64 MB and 128 MB SDRAM cases.*

Table 3-6
PCI and ADSP-21160 Addresses for 64 MB SDRAM

SD Window Register Value*	PCI 32-bit Offset From BAR4	PCI Byte Offset From BAR4	ADSP-21160 Address	Description
0x02	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x800000 – 0xBFFFFF	First 16 MB block of SDRAM
0x03	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0xC00000 – 0xFFFFFF	Second 16 MB block of SDRAM
0x00	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x1000000 – 0x13FFFFF	Third 16 MB block of SDRAM
0x01	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x1400000 – 0x17FFFFF	Fourth 16 MB block of SDRAM

* Window register values of 0x00 and 0x01 always access the last two 16 MB windows of SDRAM. Refer to section 2.3 for details.

Table 3-7
PCI and ADSP-21160 Addresses for 128 MB SDRAM

SD Window Register Value*	PCI 32-bit Offset from BAR4	PCI byte Offset from BAR4	ADSP-21160 Address	Description
0x02	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x800000 – 0xBFFFFF	First 16 MB block of SDRAM
0x03	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0xC00000 – 0xFFFFFF	Second 16 MB block of SDRAM
0x04	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x1000000 – 0x13FFFFF	Third 16 MB block of SDRAM
0x05	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x1400000 – 0x17FFFFF	Fourth 16 MB block of SDRAM
0x06	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x1800000 – 0x1BFFFFF	Fifth 16 MB block of SDRAM
0x07	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x1C00000 – 0x1FFFFF	Sixth 16 MB block of SDRAM
0x00	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x2000000 – 0x23FFFFF	Seventh 16 MB block of SDRAM
0x01	0x00000 – 0x3FFFFF	0x000000 – 0xFFFFFF	0x2400000 – 0x27FFFFF	Eighth 16 MB block of SDRAM

* Window register values of 0x00 and 0x01 always access the last two 16 MB windows of SDRAM. Refer to section 2.3 for details.

3.2 PCI Configuration Registers

Each PCI bus device contains a unique 256-byte region called the configuration header space (Figure 3-1). The configuration header is mandatory for a PCI agent to be in full compliance with the PCI Specification. The configuration header for the SharcFIN is used by the system to configure the SharcFIN's PCI interface.

This section describes each of the configuration space fields including the following:

- address
- default values
- initialization options
- bit definitions

Table 3-8 through Table 3-38 provide details on the PCI Configuration Registers.

Figure 3-1
PCI Configuration Register Memory Map

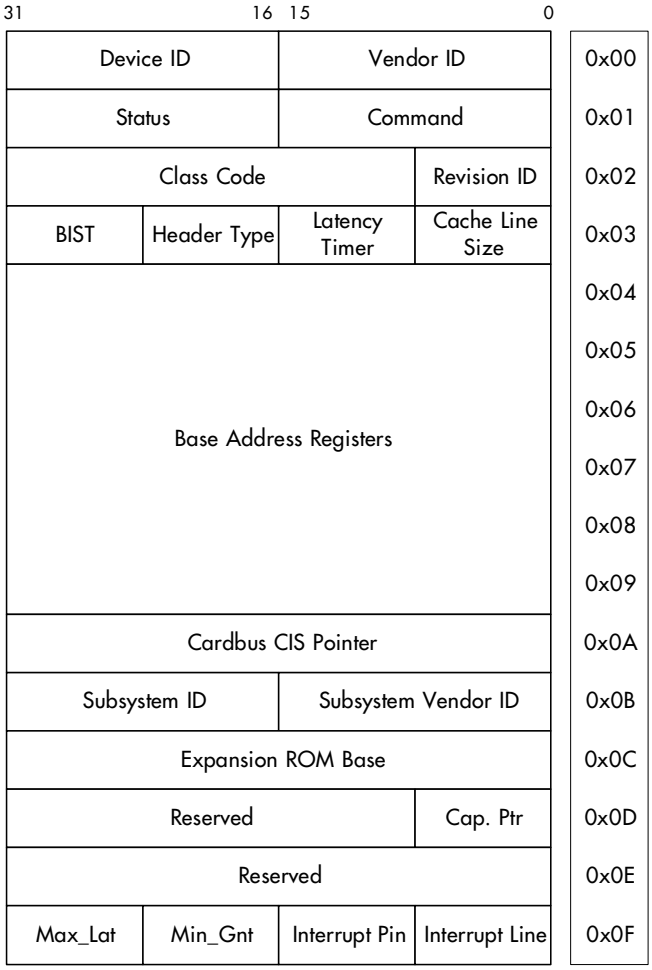


Table 3-8
Register Offset 0x00

Register Name:	PCI_ID	Register Offset: 0x00
Bits	**Function**	
31-24	*Device ID[15:8]*	
23-16	*Device ID[7:0]*	
15-8	*Vendor ID[15:8]*	
7-0	*Vendor ID[7:0]*	

Table 3-9
Device and Vendor ID Register Description

Name	Type	Reset By	Reset	Function
Device ID[15:0] Addr 0x00[31:16]	Read Only	N/A	*0x0026*	Identifies the type of SharcFIN
Vendor ID[15:0] Addr 0x00[15:0]	Read Only	N/A	*0x12BA*	Identifies the manufacturer of the device. This number is allocated by the PCI SIG to ensure its uniqueness.

Table 3-10
Register Offset 0x01

Register Name:	Register Offset: 0x01
Bits	**Function**
31-24	*Status[15:8]*
23-16	*Status[7:0]*
15-8	*Command[15:8]*
7-0	*Command[7:0]*

Table 3-11
Status Register Description

Name	Type	Reset By	Reset Value	Function
Status[15] Addr 0x01[31]	Read/ Write 1 to clear	TR*	0	Detected Parity Error 1 Parity Error Detected 0 No effect This bit is set to 1 whenever a parity error is detected. Writing a 1 to this location clears it. Writing 0 to this location has no effect.
Status[14] Addr 0x01[30]	Read/ Write 1 to clear	TR	0	Signaled System Error 1 $\overline{\textbf{SERR}}$ Asserted 0 No effect This bit is set to 1 whenever the device asserts the signal $\overline{\textbf{SERR}}$. Writing 1 to this location clears it. Writing 0 to this location has no effect.
Status[13] Addr 0x01[29]	Read/ Write 1 to clear	TR	0	Received Master Abort 1 Master Abort 0 No effect This bit is set to 1 whenever a bus master abort occurs. Writing 1 to this location clears it. Writing 0 to this location has no effect.
Status[12] Addr 0x01[28]	Read/ Write 1 to clear	TR	0	Received Target Abort 1 Initiated cycle terminated by addressed target 0 No effect This bit is set to 1 whenever this device has one of its own initiated cycles terminated by the currently addressed target. Writing 1 to this location clears it. Writing 0 to this location has no effect.
(Sheet 1 of 3)				

Table 3-11
Status Register Description (Continued)

Name	Type	Reset By	Reset Value	Function
Status[11] Addr 0x01[27]	Read/ Write 1 to clear	TR	0	**Signaled Target Abort** 1 Cycle aborted when addressed as target 0 No effect This bit is set to 1 whenever this device aborts a cycle when addressed as a target. Writing a 1 to this location clears it. Writing 0 to this location has no effect.
Status[10:9] Addr 0x01[26:25]	Read Only	N/A	01	**Device Selecting Timing** 00 Fast 01 Medium 10 Slow 11 Reserved These bits define the signal behavior of **DEVSEL** from this device when accessed as a target. The SharcFIN is a 'medium' device, so these bits are hardwired to 01.
Status[8] Addr 0x01[24]	Read/ Write 1 to clear	TR	0	**Data Parity Reported** 1 Data Parity Error Detected 0 No Effect This bit is set to 1 when a data parity error occurs for a transfer involving the device as the master. The Parity Error Enable bit, bit 6 of the Command Register, must be set in order for this bit to be set.
Status[7] Addr 0x01[23]	Read Only	N/A	*1*	**Fast Back-to-Back Capable For Target** 1 Device can accept fast back-to-back cycles as target (Hardwired)
Status[6] Addr 0x01[22]	Read Only	N/A	*0*	**User Defined Feature** This bit was used to indicate whether a device supports user-defined features. Bit was changed to RESERVED in the PCI 2.2 specification.
Status[5] Addr 0x01[21]	Read Only	N/A	*1*	**Bit 5** 1 Device is 66 MHz Capable 0 Device is 33 MHz Capable Only

(Sheet 2 of 3)

Table 3-11
Status Register Description (Continued)

Name	Type	Reset By	Reset Value	Function
Status[4] Addr 0x01[20]	Read Only	N/A	*0*	**Bit 4** 0 No New Capabilities Linked List available
Status[3:0] Addr 0x01[19:16]	Read Only	N/A	*0x0*	**Reserved**
(Sheet 3 of 3)				

* TR = Total Reset. Reset functionality is described in section 2.6.

Table 3-12
Command Register Description

Name	Type	Reset By	Reset Value	Function
Command [15:10] Addr 0x01[15:10]	Read Only	N/A	*0*	**Reserved**
Command[9] Addr 0x01[9]	Read Only	N/A	*0*	**Fast Back-to-Back Enable For Master** 0 Master not Back-to-Back capable 1 Master Back-to-Back capable This master is not back-to-back capable, so set this bit to "0".
Command[8] Addr 0x01[8]	Read/ Write	TR	*1*	**System Error Enable** Set this bit to 1 to permit the chip to drive the open drain output pin, **$\overline{\text{SERR}}$**.
Command[7] Addr 0x01[7]	Read Only	N/A	*0*	**Wait Cycle Enable** 0 Address/Data stepping disabled 1 Address/Data stepping permitted
(Sheet 1 of 3)				

Table 3-12
Command Register Description (Continued)

Name	Type	Reset By	Reset Value	Function
Command[6] Addr 0x01[6]	Read/ Write	TR	*1*	**Parity Error Enable** 0 Disabled 1 Controller checks for parity errors Setting this bit to 1 allows the controller to check for parity errors. When a data (not address) parity error is detected, the PCI bus signal **PERR** is asserted.
Command[5] Addr 0x01[5]	Read Only	N/A	*0*	**Palette Snoop Enable** 0 Palette snooping disabled 1 Palette snooping enabled This bit controls how VGA compatible and graphics devices handle access to VGA palette registers. When set to 1, palette snooping is enabled. When the bit is 0, the device should treat palette access like all other accesses. The SharcFIN cannot snoop, so this bit is set to "0".
Command[4] Addr 0x01[4]	Read Only	N/A	*0*	**Memory Write & Invalidate Enable** 0 Masters must use Memory Write command 1 Masters may use Memory Write and Invalidate PCI bus command The SharcFIN cannot generate MWI cycles so this bit is set to 0.
Command[3] Addr 0x01[3]	Read only	N/A	*0*	**Special Cycle Enable** 0 Device ignores special cycle operations 1 Device monitors special cycle operations The SharcFIN cannot respond to special cycles so this bit is set to 0.
Command[2] Addr 0x01[2]	Read/ Write	TR	*1*	**Bus Master Enable** 0 Disable bus master function 1 Enable bus master function Set to 1 to allow the device to function as a bus master.
(Sheet 2 of 3)				

Table 3-12
Command Register Description (Continued)

Name	Type	Reset By	Reset Value	Function
Command[1] Addr 0x01[1]	Read/ Write	TR	*1*	**Memory Space Enable** 0 BAR address decode disabled 1 Device responds to memory access cycles
Command[0] Addr 0x01[0]	Read/ Write	TR	*1*	**I/O Space Enable** 0 Bar address decode disabled 1 Device responds to I/O access cycles
(Sheet 3 of 3)				

Table 3-13
Register Offset 0x02

Register Name:	Register Offset: 0x02
Bits	**Function**
31-24	Class Code[23:16]
23-16	Class Code[15-8]
15-8	Class Code[7-0]
7-0	Revision ID[7:0]

Table 3-14
Class Code Register Description

Name	Type	Reset By	Reset Value	Function
Class Code[23:16] Addr 0x02[31:24]	Read Only	N/A	*0x04*	**Base Class** The SharcFIN is a multimedia device.
Class Code[15:8] Addr 0x02[23:16]	Read Only	N/A	*0x80*	**Sub-Class** The SharcFIN is not specifically an audio video device.
Class Code[7:0] Addr 0x02[8:15]	Read Only	N/A	*0x00*	**Programming Interface**

Table 3-15
Revision ID Register Description

Name	Type	Reset By	Reset Value	Function
Revision ID[7:0] Addr 0x02[7:0]	Read-Only	N/A	*0*	**Revision identification**

Table 3-16
Register Offset 0x03

Register Name:	**Register Offset: 0x03**
Bits	**Function**
1-24	*BIST[7:0]*
23-16	*Header Type[7:0]*
15-8	*Latency Timer[7:0]*
7-0	*Cache Line Size[7:0]*

Table 3-17
Built-In Self-Test (BIST) Register Description

Name	Type	Reset By	Reset Value	Function
BIST[7] Addr 0x03[31]	Read Only	N/A	*0*	1 Device is BIST capable 0 Device is not BIST capable
BIST[6] Addr 0x03[30]	Read/ Write	TR/HTR	0	**Built-In Self-Test** 1 Start BIST 0 No effect If device is BIST-capable, writing a 1 to bit 6 will commence the self test. PCI specification states that self test must terminate in 2 seconds.
BIST[5:4] Addr 0x03[29:28]	Read Only	N/A	0	**Reserved** Should be set to 0.
BIST[3:0] Addr 0x03[27:24]	Read Only	TR/HTR	0	**Result Code** 0 = BIST OK 1–15 = BIST Fail

Table 3-18
Header Type Register Description

Name	Type	Reset By	Reset Value	Function
Header Type[7] Addr 0x03[23]	Read Only	N/A	*0*	0 Single-function device 1 Multi-function device
Header Type[6:0] Addr 0x03[22:16]	Read Only	N/A	*0*	**Format Field**

Table 3-19
Latency Timer Register Description

Name	Type	Reset By	Reset Value	Function
Latency Timer[7:0] Addr 0x03[15:8]	Read/ Write	TR	0	**Latency Timer** Used only when the device is a bus master. It indicates the number of PCI bus clocks that this master will be guaranteed. May be overridden by user. (See also section entitled "Bus Mastering" on page 42.)

Table 3-20
Cache Line Size Register Description

Name	Type	Reset By	Reset Value	Function
Cache Line Size[7:0] Addr 0x03[7:0]	Read/ Write	TR	0	**Cache Line Size** This register specifies the system cache line size in units of 32-bit words. This register must be implemented by the master devices that can generate the Memory Write and Invalidate command. Slave devices that want to allow memory bursting using cache line wrap addressing mode must implement this register. Valid values are: 16 0x10 32 0x20 64 0x40 128 0x80 All other values are assumed invalid.

Table 3-21
Register Offset 0x04, 0x05, 0x06, 0x07, 0x08, 0x09

Register Name:	Register Offset: 0x04, 0x05, 0x06, 0x07, 0x08, 0x09
Bits	**Function**
31-24	*Base Address Register[31:24]*
23-16	*Base Address Register[23:16]*
15-8	*Base Address Register[15:8]*
7-0	*Base Address Register[7:0]*

Table 3-22
Base Address Register Description

Name	Type	Reset By	Reset Value	Function
	Base Address Registers (BARs) The Base Address Register (BAR) is used to assign memory or I/O space for the add-on function. There are six base address locations. The size of each base address is hard coded. The size is determined by writing all 1's to a base address register and then reading that register back. The bits returning 0's define the size of the memory region. After the size and type of decode, memory or I/O have been determined, an assigned address is written to the base address register bits returned 1's. Configuring 64-bit addressing can only be done on 0x06 and 0x08 registers. When a register is configured for 64-bit addressing, next 32-bit register is used for the upper 32 address bits of the 64-bit address match.			
Memory Configuration				
Base Address[31:4]	Read/ Write	TR	BAR0: 0xFFFFFE0 BAR1: 0xFFC0000 BAR2: 0xFE00000 BAR3: 0xFFFFFFF BAR4: 0xFF00000 BAR5: 0xFFFFFFF	**Base Address** Hard coded for BAR0 = 512 Bytes BAR1 = 4 MBytes BAR2 = 32 MBytes BAR4 = 16 MBytes
Base Address[3]	Read Only	TR	BAR0: 0 BAR1: 0 BAR2: 1 BAR3: 1 BAR4: 1 BAR5: 1	**Prefetchable** This byte is set to 1 if there are no side effects on reads, the device returns all bytes on reads regardless of the byte enables.
(Sheet 1 of 2)				

Table 3-22
Base Address Register Description (Continued)

Name	Type	Reset By	Reset Value	Function
Base Address[2:1]	Read Only	TR	BAR0: 00 BAR1: 00 BAR2: 10 BAR3: 11 BAR4: 10 BAR5: 11	00 Locate anywhere in 32-bit address 01 Below 1 MB 10 Locate anywhere in 64-bit address 11 Reserved
Base Address[0]	Read Only	TR	BAR0: 0 BAR1: 0 BAR2: 0 BAR3: 1 BAR4: 0 BAR5: 1	**Memory vs. I/O Space Configuration** 0 Memory Space Configuration 1 I/O Space Configuration
(Sheet 2 of 2)				

Table 3-23
Register Offset 0x0A

Register Name:	**Register Offset: 0x0A**
Bits	**Function**
31-24	*Cardbus CIS Pointer[31:24]*
23-16	*Cardbus CIS Pointer[23:16]*
15-8	*Cardbus CIS Pointer[15:8]*
7-0	*Cardbus CIS Pointer[7:0]*

Table 3-24
Cardbus CIS Pointer Register Description

Name	Type	Reset By	Reset Value	Function
Cardbus CIS Pointer[31:0] Addr 0x0A[31:0]	Read Only	N/A	*0x00000000*	**Cardbus CIS Pointer** Cardbus CIS pointer is set to 0x0. For a detailed explanation of the CIS, refer to the PCMCIA V2.10 specification. The subject is covered under the heading of "Card Metaformat" and describes the type of information provided and the organization of this information.

Table 3-25
Register Offset 0x0B

Register Name:	Register Offset: 0x0B
Bits	**Function**
31-24	*Subsystem ID[15:8]*
23-16	*Subsystem ID[7:0]*
15-8	*Subsystem Vendor ID[15:8]*
7-0	*Subsystem Vendor ID[7:0]*

Table 3-26
Subsystem ID/Subsystem Vendor ID Register Description

Name	Type	Reset By	Reset Value	Function
Subsystem ID[15:0] Addr 0x0B[31:16]	Read Only	N/A	0x0000	These optional registers are used to uniquely identify the add-in board or system where the PCI device resides. They provide a mechanism for add-in card vendors to distinguish their cards from one another even though the cards may have the same *Vendor ID* and *Device ID*.
Subsystem Vendor ID[15:0] Addr 0x0B[15:0]	Read Only	N/A	0x0000	

Table 3-27
Register Offset 0x0C

Register Name:	Register Offset: 0x0C
Bits	**Function**
31-24	*Expansion ROM Base Address[31:24]*
23-16	*Expansion ROM Base Address[23:16]*
15-8	*Expansion ROM Base Address[15:8]*
7-0	*Expansion ROM Base Address[7:0]*

Table 3-28
Expansion ROM Base Address Register Description

Name	Type	Reset By	Reset Value	Function
Expansion ROM Base Address[31:11] Addr 0x0C[31:11]	Read Only	N/A	*0x0*	**Base Address** The SharcFIN does not use expansion ROM.
Expansion ROM Base Address[10:1] Addr 0x0C[10:1]	Read Only	N/A	0	**Reserved** (Always reads as "0")
Expansion ROM Base Address[0] Addr 0x0C[0]	Read Only	N/A	*0x0*	**Address Decode Enable** 0 Address Decode Disabled 1 Address Decode Enabled

Table 3-29
Register Offset 0x0D

Register Name:	Register Offset: 0x0D
Bits	**Function**
31-24	*Reserved[23:16]*
23-16	*Reserved[15:8]*
15-8	*Reserved[7:0]*
7-0	*Cap_Ptr[7:0]*

Table 3-30
Reserved Register Description

Name	Type	Reset By	Reset Value	Function
Reserved[23:0] Addr 0x0D[31:8]	Read Only	N/A	*0x000000*	Reserved registers must return 0 on read as specified in the PCI specification.

Table 3-31
Capabilities Pointer Register Description

Name	Type	Reset By	Reset Value	Function
Cap_Ptr[7:0] Addr 0x0D[7:0]	Read Only	N/A	*0x00*	Pointer to linked list of new capabilities.

Table 3-32
Register Offset 0x0E

Register Name:	Register Offset: 0x0E
Bits	**Function**
31-24	*Reserved[31:24]*
23-16	*Reserved[23:16]*
15-8	*Reserved[15:8]*
7-0	*Reserved[7:0]*

Table 3-33
Reserved Register Description

Name	Type	Reset By	Reset Value	Function
Reserved[31:0] Addr 0x0E[31:0]	Read Only	N/A	*0x00000000*	Reserved registers must return 0 on read as specified in the PCI specification.

Table 3-34
Register Offset 0x0F

Register Name:	Register Offset: 0x0F
Bits	**Function**
31-24	*Max_Lat[7:0]*
23-16	*Min_Gnt[7:0]*
15-8	*Interrupt Pin[7:0]*
7-0	*Interrupt Line[7:0]*

Table 3-35
Maximum Latency Register Description

Name	Type	Reset By	Reset Value	Function
Max_Lat[7:0] Addr 0x0F[31:24]	Read Only	N/A	*0x00*	**Maximum Latency** This register specifies how often the device needs to gain access to the PCI bus. Setting this register to 0 indicates that the device has no major requirements for the settings of Latency Timers.

Table 3-36
Minimum Grant Register Description

Name	Type	Reset By	Reset Value	Function
Min_Gnt[7:0] Addr 0x0F[23:16]	Read Only	N/A	*0x00*	**Minimum Grant** This register specifies how long of a burst period the device needs assuming a clock rate of 33 MHz.

Table 3-37
Interrupt Pin Register Description

Name	Type	Reset By	Reset Value	Function
Interrupt Pin[7:0] Addr 0x0F[15:8]	Read Only	N/A	*0x01*	Defines which of the four PCI interrupt request pins the device uses.

Table 3-38
Interrupt Line Register Description

Name	Type	Reset By	Reset Value	Function
Interrupt Line [7:0] Addr 0x0F[7:0]	Read/ Write	TR	0	Eight-bit register indicating interrupt line routing information.

3.3 Chip Control Registers

The SharcFIN contains a 512-byte structure that houses the control registers. All the operations of the SharcFIN are controlled using these registers. The control structure can be accessed either via the ADSP-21160 cluster bus or via the PCI bus. The control registers are at an offset of 0x00–0x17F to ***Base Address Register 0*** (BAR0) or MS2 base + 0x00 to 0x5F (see Figure 3-2 and Figure 3-3, and Table 3-39 through Table 3-171).

Figure 3-2
PCI Control Register Memory Map

Figure 3-3
PCI Control Register Memory Map (Continued)

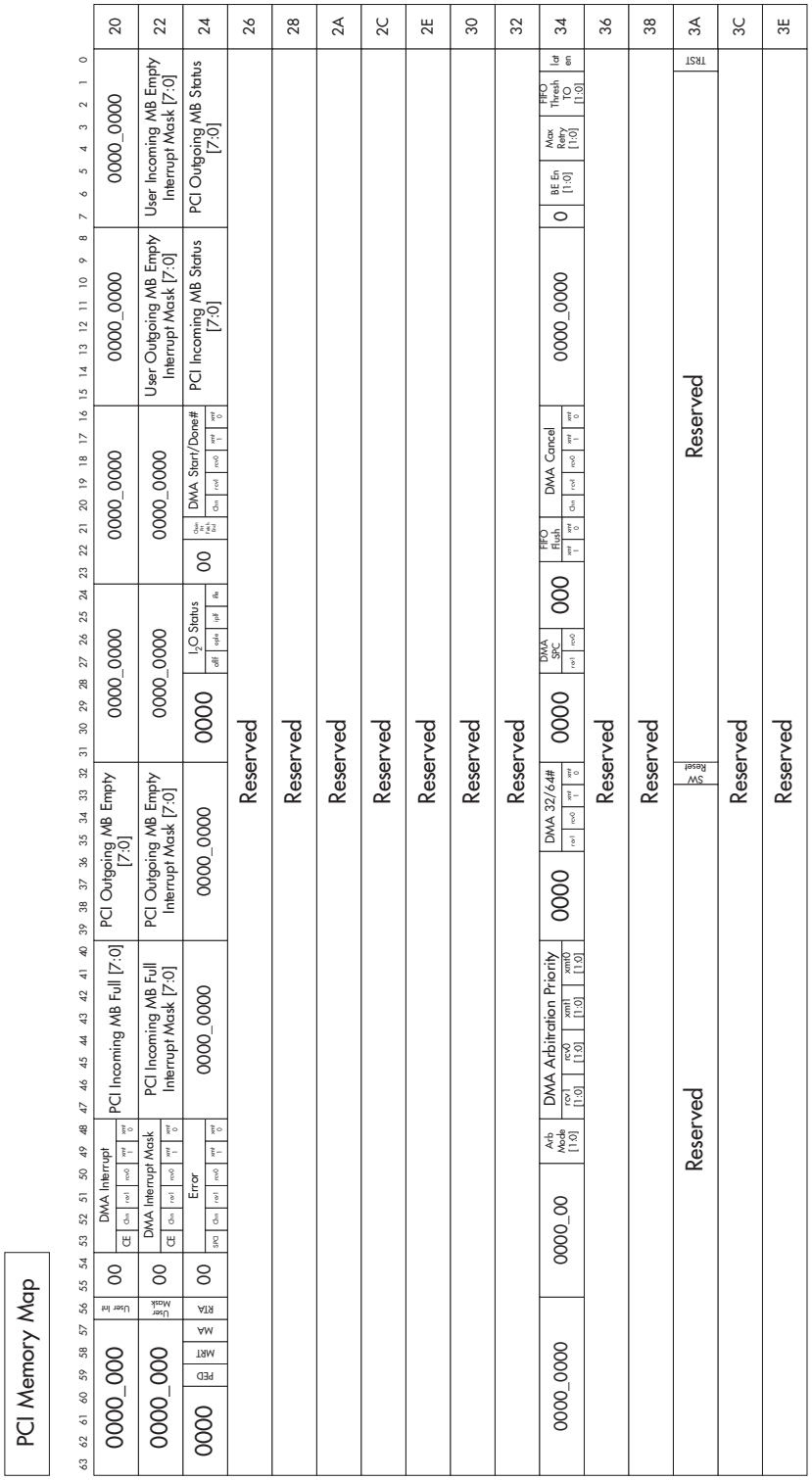


Figure 3-4
User Control Register Memory Map (Continues on next page)

Figure 3-5
User Control Register Memory Map (Continued)

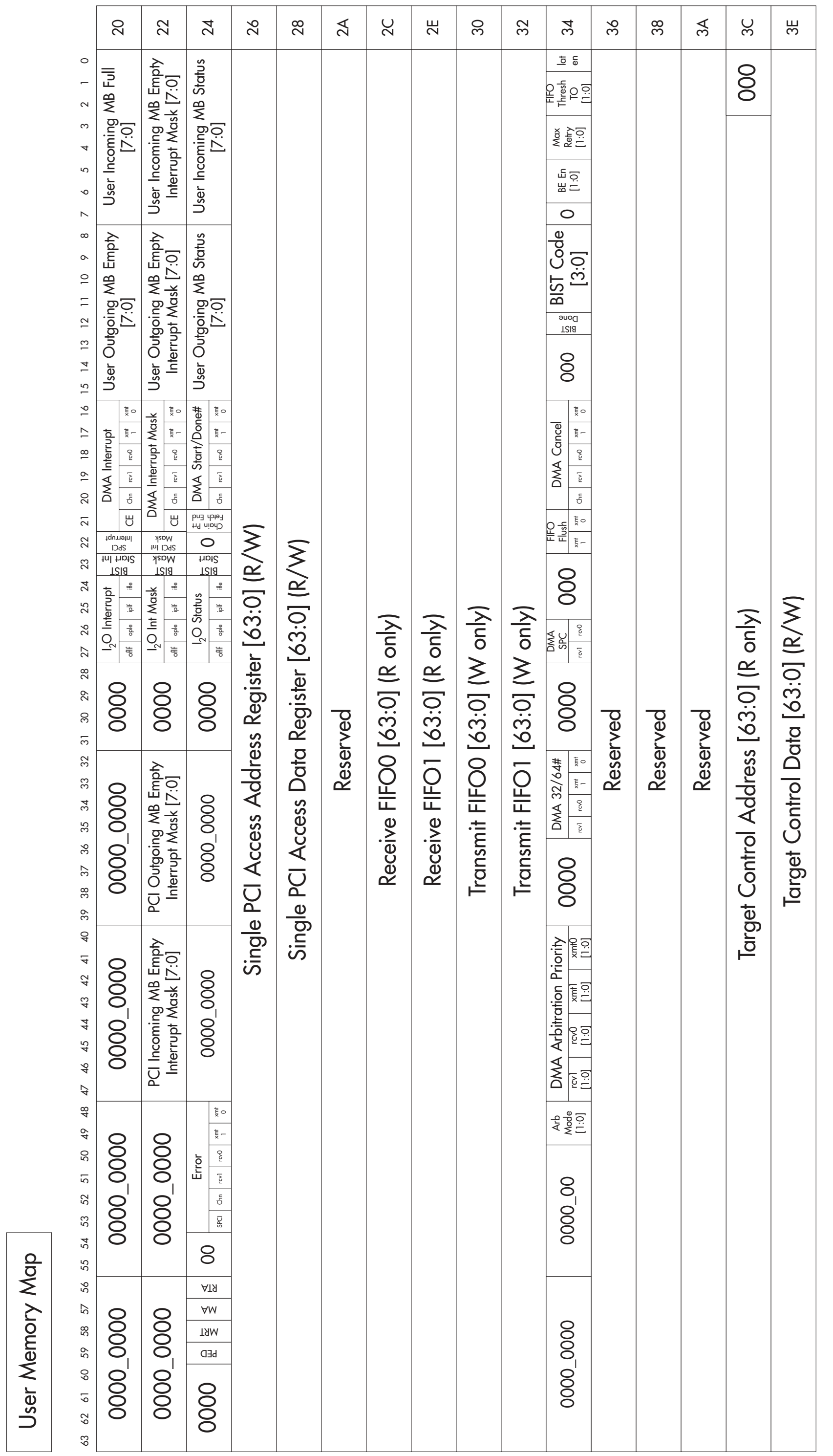
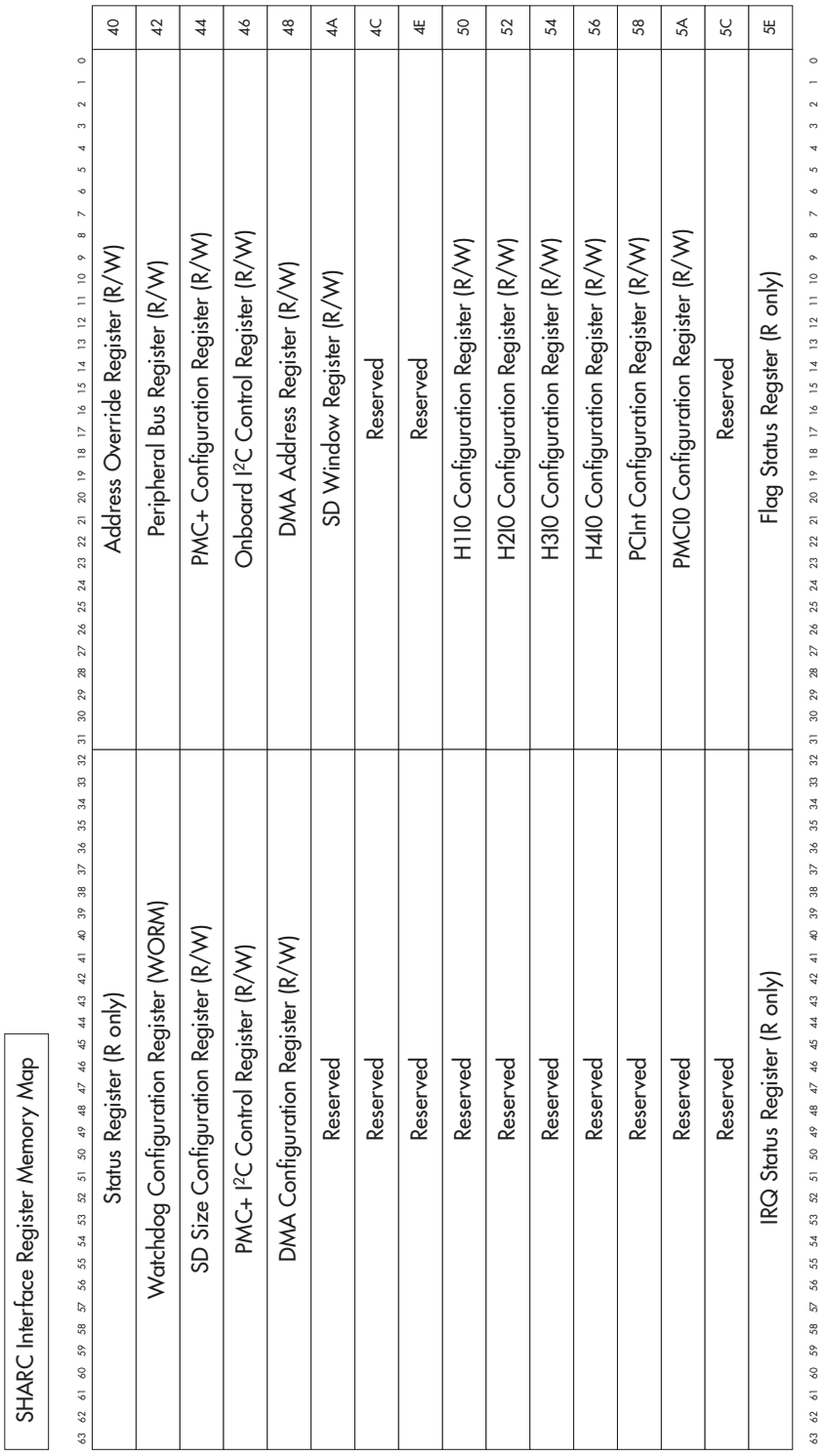


Figure 3-6
SHARC Interface Control Register Memory Map



3.3.1 PCI Control Registers

All access to the control registers via the PCI bus are synchronized to **LClk** and simultaneous accesses to the same byte lane of data result in the PCI write being dropped. (Exceptions apply to **DMA Start/Done#**, **DMA Cancel**, and **Transmit FIFO Flush**). The control registers may be accessed via the ADSP-21160 cluster bus on a clock-by-clock basis.

Table 3-39
Control Register Offset 0x00

Register Name: PCI_ID Register Offset: 0x00	
Bits	**Function**
63-56	*Master Write Address 0[63:56]*
55-48	*Master Write Address 0[55:48]*
47-40	*Master Write Address 0[47:40]*
39-32	*Master Write Address 0[39:32]*
31-24	*Master Write Address 0[31:24]*
23-16	*Master Write Address 0[23:16]*
15-8	*Master Write Address 0[15:8]*
7-0	*Master Write Address 0[7:0]*

Table 3-40
Master Write Address 0 Description

Name	**Type**	**Reset By**	**Reset Value**	**Function**
Master Write Address 0[63:0] addr 0x00[63:0]	Read/ Write	TR/HTR*	*0*	Establishes the starting PCI target address for data moving from Transmit FIFO0 to the PCI bus under the control of Transmit DMA Channel 0. The address must be 64-bit aligned. A simultaneous write from both busses results in the PCI write being dropped.

* TR= Total Reset
HTR = Host Total Reset
Reset functionality is described in Section 2.6.

Table 3-41
Control Register Offset 0x02

Register Name: PCI_ID Register Offset: 0x02	
Bits	**Function**
63-56	*Master Write Count Status 0[31:24]*
55-48	*Master Write Count Status 0[23:16]*
47-40	*Master Write Count Status 0[15:8]*
39-32	*Master Write Count Status 0[7:0]*
31-24	*Master Write Transfer Count 0[31:24]*
23-16	*Master Write Transfer Count 0[23:16]*
15-8	*Master Write Transfer Count 0[15:8]*
7-0	*Master Write Transfer Count 0[7:0]*

Table 3-42
Master Write Count Status 0 Description

Name	Type	Reset By	Reset Value	Function
Master Write Count Status 0[31:0] addr 0x02[63:32]	Read Only	TR/HTR	0	The status of the transfer operation for Transmit DMA Channel 0 is monitored by reading this register. The ***Master Write Count Status 0*** register will contain the number of bytes yet to be transferred (or the last value after a cancel) if the Transmit DMA Channel 0 is enabled. Otherwise the register will read 0. This register is accessible from both the PCI and the ADSP-21160 cluster bus.

Table 3-43
Master Write Transfer Count 0 Description

Name	Type	Reset By	Reset Value	Function
Master Write Transfer Count 0[31:0] addr 0x02[31:0]	Read/ Write	TR/HTR	0	Total number of bytes that are to be transferred from Transmit FIFO0 to the PCI bus by Transmit DMA Channel 0. Transfers may be any non-zero value up to 0xFFFFFFF0 (2^{32}–16). This register is read/write and may be accessed from both the ADSP-21160 cluster bus and by PCI. A simultaneous write from both busses results in the PCI write being dropped.

Table 3-44
Control Register Offset 0x04

Register Name: PCI_ID Register Offset: 0x04	
Bits	**Function**
63-56	*Master Write Address 1[63:56]*
55-48	*Master Write Address 1[55:48]*
47-40	*Master Write Address 1[47:40]*
39-32	*Master Write Address 1[39:32]*
31-24	*Master Write Address 1[31:24]*
23-16	*Master Write Address 1[23:16]*
15-8	*Master Write Address 1[15:8]*
7-0	*Master Write Address 1[7:0]*

Table 3-45
Master Write Address 1 Description

Name	Type	Reset By	Reset Value	Function
Master Write Address 1[63:0] addr 0x04[63:0]	Read/ Write	TR/HTR	*0*	Establishes the starting PCI target address for data moving from Transmit FIFO1 to the PCI bus under the control of Transmit DMA Channel 1, and gets sampled by the DMA engine upon start of the transfer. This register does not get updated during the transfer. The address must be aligned on 64-bit boundaries. A simultaneous write from both busses results in the PCI write being dropped.

Table 3-46
Control Register Offset 0x06

Register Name: PCI_ID Register Offset: 0x06	
Bits	**Function**
63-56	*Master Write Count Status 1[31:24]*
55-48	*Master Write Count Status 1[23:16]*
47-40	*Master Write Count Status 1[15:8]*
39-32	*Master Write Count Status 1[7:0]*
31-24	*Master Write Transfer Count 1[31:24]*
23-16	*Master Write Transfer Count 1[23:16]*
15-8	*Master Write Transfer Count 1[15:8]*
7-0	*Master Write Transfer Count 1[7:0]*

Table 3-47
Master Write Count Status 1 Description

Name	Type	Reset By	Reset Value	Function
Master Write Count Status 1[31:0] addr 0x06[63:32]	Read Only	TR/HTR	*0*	The status of the transfer operation for Transmit DMA Channel 1 is monitored by reading this register. The ***Master Write Count Status 1*** register will contain the number of bytes yet to be transferred if the *Transmit DMA Channel 1* is enabled. This register is updated approximately every eight clock cycles. A maskable interrupt can be generated to either the PCI bus or the ADSP-21160 cluster bus upon completion. This register is accessible from both the PCI and the ADSP-21160 cluster bus.

Table 3-48
Master Write Transfer Count 1 Description

Name	Type	Reset By	Reset Value	Function
Master Write Transfer Count 1[31:0] addr 0x06[31:0]	Read/ Write	TR/HTR	*0*	Total number of bytes that are to be transferred from Transmit FIFO 1 to the PCI bus by Transmit DMA Channel 1. Transfers may be any non-zero value up to 0xFFFFFFF0 (2^{32}–16). This register is read/write and may be accessed from both the ADSP-21160 cluster bus and by PCI. A simultaneous write from both busses results in the PCI write being dropped. This register gets sampled at the start of a DMA transfer.

Table 3-49
Control Register Offset 0x08

Register Name: PCI_ID Register Offset: 0x08								
Bits	**Function**							
63-56	*00*		*Single PCI Start (user only)*	*SPCI 32-bit (user only)*	*SPCI Command[3:0] [User Only]*			
55-48	*Single PCI Access Byte Lane[7:0] [User Only]*							
47-40	*Receive FIFO1 Byte Lane[7:0] [User Only]*							
39-32	*Bus Request [3:0]*				*Pipeline Not Empty[1:0]*		*Chain tag0[1:0] [User Only]*	
31-24	*Receive FIFO0 Byte Lane[7:0] [User Only]*							
23-16	*XMT1 Full*	*XMT1 Almost Full*	*XMT0 Full*	*XMT0 Almost Full*	*RCV1 Empty*	*RCV1 Almost Empty*	*RCV0 Empty*	*RCV0 Almost Empty*
15-8	*Chip Revision ID[7:0] (read only)*							
7-0	*User ID[7:0] (read only)*							

Table 3-50
Single PCI Access Start Description

Name	Type	Reset By	Reset Value	Function
Single PCI Access Start addr 0x08[61]	Read/ Write (user)	TR/HTR	*0*	**Single PCI Access Start/Status** On write: 1 Start SPCI access 0 No effect On read: 1 SPCI access not complete 0 SPCI access complete. May be accessed from the ADSP-21160 cluster bus only.

Table 3-51
Single PCI Access 32-Bit Description

Name	Type	Reset By	Reset Value	Function
Single PCI Access 32-Bit addr 0x08[60]	Read/ Write (user)	TR/HTR	0	**Automatic/Predetermined Target Bus Width for SPCI Access:** 1 Assume destination is 32 bits. 0 Automatically determine destination bus width using **$\overline{\text{REQ64}}$.** Initially assume destination is 64-bits. Only has meaning on memory accesses; otherwise ignored. May be accessed from the ADSP-21160 cluster bus only.

Table 3-52
Single PCI Access Command Description

Name	Type	Reset By	Reset Value	Function
Single PCI Access Command[3:0] addr 0x08[59:56]	Read/ Write (user)	TR/HTR	0	**SPCI Command:** **C/ Command Type** 0000 Interrupt Acknowledge 0001 Special Cycle 0010 I/O Read 0011 I/O Write 0100 Reserved 0101 Reserved 0110 Memory Read 0111 Memory Write 1000 Reserved 1001 Reserved 1010 Configuration Read 1011 Configuration Write 1100 Memory Read Multiple 1101 Dual Address Cycle 1110 Memory Read Line 1111 Memory Write and May be accessed from the ADSP-21160 cluster bus only

Table 3-53
Single PCI Byte Lanes Description

Name	Type	Reset By	Reset Value	Function
Single PCI Access Byte Lanes[7] addr 0x08[55]	Read/ Write (user)	TR/HTR	0	SPCI Data [63:56]
Single PCI Access Byte Lanes[6] addr 0x08[54]				SPCI Data [55:48]
Single PCI Access Byte Lanes[5] addr 0x08[53]				SPCI Data [47:40]
Single PCI Access Byte Lanes[4] addr 0x08[52]				SPCI Data [39:32]
Single PCI Access Byte Lanes[3] addr 0x08[51]				SPCI Data [31:24]
Single PCI Access Byte Lanes[2] addr 0x08[50]				SPCI Data [23:16]
Single PCI Access Byte Lanes[1] addr 0x08[49]				SPCI Data [15:8]
Single PCI Access Byte Lanes[0] addr 0x08[48]				SPCI Data [7:0] 1 Active Byte Lane 0 Inactive Byte Lane May be accessed from the ADSP-21160 cluster bus only

Table 3-54
Receive FIFO1 Byte Lane Description

Name	Type	Reset By	Reset Value	Function
Receive FIFO1 Byte Lane[7] addr 0x08[47]	Read Only (user)	TR/HTR	*0*	Byte lane for the data present at the output of Receive FIFO1 [63:56].
Receive FIFO1 Byte Lane[6] addr 0x08[46]				Byte lane for the data present at the output of Receive FIFO1 [55:48].
Receive FIFO1 Byte Lane[5] addr 0x08[45]				Byte lane for the data present at the output of Receive FIFO1 [47:40].
Receive FIFO1 Byte Lane[4] addr 0x08[44]				Byte lane for the data present at the output of Receive FIFO1 [39:32].
Receive FIFO1 Byte Lane[3] addr 0x08[43]				Byte lane for the data present at the output of Receive FIFO1 [31:24].
Receive FIFO1 Byte Lane[2] addr 0x08[42]				Byte lane for the data present at the output of Receive FIFO1 [23:16].
Receive FIFO1 Byte Lane[1] addr 0x08[41]				Byte lane for the data present at the output of Receive FIFO1 [15:8].
Receive FIFO1 Byte Lane[0] addr 0x08[40]				Byte lane for the data present at the output of Receive FIFO1 [7:0]. 1 Active Byte Lane 0 Inactive Byte Lane May be accessed from the ADSP-21160 cluster bus only

Table 3-55
Bus Request Status Description

Name	Type	Reset By	Reset Value	Function
Bus Request[3] addr 0x08[39]	Read Only	TR/HTR	*0*	**Transmit DMA Channel 0 Activity Status** 1 Transmit DMA Channel 0 has pending need of the PCI bus 0 No pending transactions
Bus Request[2] addr 0x08[38]	Read Only	TR/HTR	0	**Transmit DMA Channel 1 Activity Status** 1 Transmit DMA Channel 1 has pending need of the PCI bus 0 No pending transactions
Bus Request[1] addr 0x08[37]	Read Only	TR/HTR	0	**Receive DMA Channel 0 Activity Status** 1 Receive DMA Channel 0 has pending need of the PCI bus 0 No pending transactions
Bus Request[0] addr 0x08[36]	Read Only	TR/HTR	0	**Receive DMA Channel 1 Activity Status** 1 Receive DMA Channel 1 has pending need of the PCI bus 0 No pending transactions

Table 3-56
Pipeline Not Empty Description

Name	Type	Reset By	Reset Value	Function
Pipeline Not Empty[1] addr 0x08[34]	Read Only	TR/HTR	*0*	**Transmit 1 Pipeline Status** Used to determine when the Transmit pipeline is empty for gracefully cancelling the DMA operation. 1 Pipeline not empty 0 Pipeline is empty
Pipeline Not Empty[0] addr 0x08[35]	Read Only	TR/HTR	0	**Transmit 0 Pipeline Status** Used to determine when the Transmit pipeline is empty for gracefully cancelling the DMA operation. 1 Pipeline not empty 0 Pipeline is empty

Table 3-57
Chain Tag Field Description

Name	Type	Reset By	Reset Value	Function
Chain tag0[1:0] addr 0x08[33:32]	Read Only (user)	TR/ HTR	*0*	**Chain Tag[1:0]** 00 Normal Data 01 Descriptor Pointer quadword 0 (PCI StartingAddress) 10 Descriptor Pointer quadword 1 (User Defined) 11 Descriptor Pointer quadword 2 (Transfer Count, et. al.) May be accessed from the ADSP-21160 cluster bus only.

Table 3-58
Receive FIFO0 Byte Lane Description

Name	Type	Reset By	Reset Value	Function
Receive FIFO0 Byte Lane[7] addr 0x08[31] *Receive FIFO0 Byte Lane[6]* addr 0x08[30] *Receive FIFO0 Byte Lane[5]* addr 0x08[29] *Receive FIFO0 Byte Lane[4]* addr 0x08[28] *Receive FIFO0 Byte Lane[3]* addr 0x08[27] *Receive FIFO0 Byte Lane[2]* addr 0x08[26] *Receive FIFO0 Byte Lane[1]* addr 0x08[25] *Receive FIFO0 Byte Lane[0]* addr 0x08[24]	Read Only (user)	TR/HTR	*0*	Byte lane for the data present at the output of Receive FIFO0 [63:56]. Byte lane for the data present at the output of Receive FIFO0 [55:48]. Byte lane for the data present at the output of Receive FIFO0 [47:40]. Byte lane for the data present at the output of Receive FIFO0 [39:32]. Byte lane for the data present at the output of Receive FIFO0 [31:24]. Byte lane for the data present at the output of Receive FIFO0 [23:16]. Byte lane for the data present at the output of Receive FIFO0 [15:8]. Byte lane for the data present at the output of Receive FIFO0 [7:0]. 1 Active Byte Lane 0 Inactive Byte Lane May be accessed from the ADSP-21160 cluster bus only

Table 3-59
Transmit Descriptions

Name	Type	Reset By	Reset Value	Function
XMT1 Full FIFO Flag addr 0x08[23]	Read Only	TR/HTR	*0*	**Transmit FIFO1 Full Flag** Active High. Full flag for Transmit FIFO1. Indicates that the FIFO is full. Identical to the signal **xmt1_fifo_ff**, except for a one pipe delay.
XMT1 Almost Full FIFO Flag addr 0x08[22]	Read Only	TR/HTR	0	**Transmit FIFO1 Almost Full Flag** Active High. Almost full flag for Transmit FIFO1. Indicates that the number of quadwords remaining in Transmit FIFO1 is greater than or equal to the threshold programmed in register 0x1A bits[61:56]. Identical to the signal **xmt1_fifo_program_full_flag**, except for a one pipe delay.
XMT0 Full FIFO Flag addr 0x08[21]	Read Only	TR/HTR	0	**Transmit FIFO0 Full Flag** Active High. Full flag for Transmit FIFO0. Indicates that the FIFO is full. Identical to the signal **xmt0_fifo_ff** except for a one pipe delay.
XMT0 Almost Full FIFO Flag addr 0x08[20]	Read Only	TR/HTR	0	**Transmit FIFO0 Almost Full Flag** Active High. Almost full flag for DMA Transmit FIFO0. Indicates that the number of quadwords remaining in Transmit FIFO0 is greater than or equal to the threshold programmed in register 0x1A bits[53:48]. Identical to the signal **xmt0_fifo_program_full_flag** except for a one pipe delay.

Table 3-60
Receive Descriptions

Name	Type	Reset By	Reset Value	Function
RCV1 Empty FIFO Flag addr 0x08[19]	Read Only	TR/HTR	*0*	**Receive FIFO1 Empty Flag** Active High. Empty flag for Receive FIFO1. Indicates that the FIFO is empty. Identical to the signal **rcv1_fifo_ef** except for a one pipe delay.
RCV1 Almost Empty FIFO Flag addr 0x08[18]	Read Only	TR/HTR	0	**Receive FIFO1 Almost Empty Flag** Active High. Almost empty flag for Receive FIFO1. Indicates that the number of quadwords remaining in Receive FIFO1 is equal to or less than the threshold programmed in register 0x1A bits[13:8]. Identical to the signal **rcv1_fifo_program_empty_flag** except for a one pipe delay.
RCV0 Empty FIFO Flag addr 0x08[17]	Read Only	TR/HTR	0	**Receive FIFO0 Empty Flag** Active High. Empty flag for Receive FIFO0. Indicates that the FIFO is empty. Identical to the signal **rcv0_fifo_ef** except for a one pipe delay.
RCV0 Almost Empty FIFO Flag addr 0x08[16]	Read Only	TR/HTR	0	**Receive FIFO0 Almost Empty Flag** Active High. Almost empty flag for Receive FIFO0. Indicates that the number of quadwords remaining in Receive FIFO0 is equal to or less than the threshold programmed in register 0x1A bits[5:0]. Identical to the signal **rcv0_fifo_program_empty_flag** except for a one pipe delay.

Table 3-61
Chip Revision ID Description

Name	Type	Reset By	Reset Value	Function
Chip Revision ID[7:0] addr 0x08[15:8]	Read Only	N/A	*01*	**Chip Revision** Set by factory. Should read 0x01 for revision 1 silicon.

Table 3-62
User ID Description

Name	Type	Reset By	Reset Value	Function
User ID[7:0] addr 0x08[7:0]	Read Only	N/A	*0x05*	**SharcFIN Revision ID**

Table 3-63
Control Register Offset 0x0A

Register Name: PCI_ID Register Offset: 0x0A						
Bits	**Function**					
63-56	*0000*		*Enable 16/8 Target Timeout*	*Wait Timeout Read*	*Wait Timeout Write*	*Bar Enable ROM*
55-48	*BAR Enable[5:0]*				*Target BAR Configuration[17:16]*	
47-40	*Target BAR Configuration[15:8]*					
39-32	*Target BAR Configuration[7:0]*					
31-24	*Target FIFO Threshold MSBs[3:0]*		*Target FIFO Control–Emptiness Threshold[11:8]*			
23-16	*Target FIFO Control–Emptiness Threshold[7:0]*					
15-8	*0*	*Target Prefetch Control[6:0]*				
7-0	*0*	*Target Burst Request[6:0]*				

Table 3-64
Enable 16/8 Target Timeout Readback Description

Name	Type	Reset By	Reset Value	Function
Antifuse Status: Enable16/8 Target Timeout addr 0x0A[59]	Read Only	N/A	*1*	**Target 16/8 Timeout Enable Readback** 1 Enable target termination to generate **STOP#** at 16 *pci_clk_2fpgas* after **FRAME#** or after8 *pci_clk_2fpgas* after each data phase during a burst. 0 Disable the above timeout.

Table 3-65
Wait Timeout Read Readback Description

Name	Type	Reset By	Reset Value	Function
Antifuse Status: Wait Timeout Read addr 0x0A[58]	Read Only	N/A	*1*	**Target Wait Timeout on Read Readback** 1 Wait 16/8 timeout (or infinite) until data is available 0 Target drives **STOP** immediately when it can't assert **TRDY** on a read access.

Table 3-66
Wait Timeout Write Readback Description

Name	Type	Reset By	Reset Value	Function
Wait Timeout Write addr 0x0A[57]	Read Only	N/A	*1*	**Target Wait Timeout on Write Readback** 1 Wait for 16/8 timeout (or infinite) until space is available in the Target Write/ Post FIFO. 0 Target drives **STOP#** immediately when it can't assert **TRDY#** on a write access.

Table 3-67
BAR Enable Readback Description

Name	Type	Reset By	Reset Value	Function
BAR Enable ROM addr 0x0A[56]	Read Only	N/A	*0*	**ROM BIOS Base Address Register Enable Readback** 1 Enabled 0 Disabled
BAR Enable[5] addr 0x0A[55]	Read Only	N/A	*0*	**Base Address Register 5 Enable Readback** 1 Enabled 0 Disabled
BAR Enable[4] addr 0x0A[54]	Read Only	N/A	*1*	**Base Address Register 4 Enable Readback** 1 Enabled 0 Disabled
BAR Enable[3] addr 0x0A[53]	Read Only	N/A	*0*	**Base Address Register 3 Enable Antifuse Readback** 1 Enabled 0 Disabled
BAR Enable[2] addr 0x0A[52]	Read Only	N/A	*1*	**Base Address Register 2 Enable Readback** 1 Enabled 0 Disabled
BAR Enable[1] addr 0x0A[51]	Read Only	N/A	*1*	**Base Address Register 1 Enable Readback** 1 Enabled 0 Disabled
BAR Enable[0] addr 0x0A[50]	Read Only	N/A	*1*	**Base Address Register 0 Enable Readback** 1 Enabled 0 Disabled

Table 3-68
Target BAR Configuration Description

Name	Type	Reset By	Reset Value	Function
Target BAR Configuration	Read Only	N/A		**Bar Configuration Antifuse Status** 3 — 1 Prefetchable 0 Not Prefetchable 2 — 1 64-Bit Bar 0 32-Bit Bar 0 — 1 I/O Space 0 Memory Space
BAR[5]—Bits[17:15] addr 0x0A[49:47]			*111*	
BAR[4]—Bits[14:12] addr 0x0A[46:44]			*110*	
BAR[3]—Bits[11:9] addr 0x0A[43:41]			*111*	
BAR[2]—Bits[8:6] addr 0x0A[40:38]			*110*	
BAR[1]—Bits[5:3] addr 0x0A[37:35]			*000*	
BAR[0]—Bits[2:0] addr 0x0A[34:32]			*000*	

Table 3-69
Target FIFO Threshold MSBs Description

Name	Type	Reset By	Reset Value	Function
Target FIFO Threshold MSB[3:0] addr 0x0A[31:28]	Read/ Write	TR/HTR	0	Number of empty positions necessary for Target Write/Post FIFO to accept target write transactions. Hex = Quadword spaces required 0 = 2 1 = 2 2 = 4 3 = 6 4 = 8 5 = 10 6 = 12 7 = 14 8 = 16 9 = 18 a = 20 b = 22 c = 24 d = 26 e = 28 f = 30 (Valid for BARs with Emptiness Threshold Control set to "10")

Table 3-70
Target FIFO Control—Emptiness Threshold

Name	Type	Reset By	Reset Value	Function
Emptiness Threshold Control[11:0] ***BAR[5]***—Bits[11:10] addr 0x0A[27:26] ***BAR[4]***—Bits[9:8] addr 0x0A[25:24] ***BAR[3]***—Bits[7:6] addr 0x0A[23:22] ***BAR[2]***—Bits[5:4] addr 0x0A[21:20] ***BAR[1]***—Bits[3:2] addr 0x0A[19:18] ***BAR[0]***—Bits[1:0] addr 0x0A[17:16]	Read/ Write	TR/HTR	0	**BAR[5] Emptiness Threshold Control** **BAR[4] Emptiness Threshold Control** **BAR[3] Emptiness Threshold Control** **BAR[2] Emptiness Threshold Control** **BAR[1] Emptiness Threshold Control** **BAR[0] Emptiness Threshold Control** Number of positions that must be available in the **Target Write/Post FIFO** for the SharcFIN to accept a target write: 00 2 or more 01 3 or more 10 Use threshold set in *Target FIFO Threshold MSB*[3:0] register 11 FIFO must be empty

Table 3-71
Target Prefetch Control Description

Name	Type	Reset By	Reset Value	Function
Target Prefetch Control[6] addr 0x0A[14]	Read/ Write	TR/ HTR	0	**ROM BAR Prefetch Control**
Target Prefetch Control[5] addr 0x0A[13]				**BAR[5] Prefetch Control**
Target Prefetch Control[4] addr 0x0A[12]				**BAR[4] Prefetch Control**
Target Prefetch Control[3] addr 0x0A[11]				**BAR[3] Prefetch Control**
Target Prefetch Control[2] addr 0x0A[10]				**BAR[2] Prefetch Control**
Target Prefetch Control[1] addr 0x0A[9]				**BAR[1] Prefetch Control**
Target Prefetch Control[0] addr 0x0A[8]				**BAR[0] Prefetch Control** Prefetch enable on burst read request from PCI 1 Allow Prefetching 0 Disallow Prefetching

Table 3-72
Target Burst Request Description

Name	Type	Reset By	Reset Value	Function
Target Burst Request[6] addr 0x0A[6]	Read/ Write	TR/HTR	*0*	**ROM BAR Burst Request**
Target Burst Request[5] addr 0x0A[5]				**BAR5 Burst Request**
Target Burst Request[4] addr 0x0A[4]				**BAR4 Burst Request**
Target Burst Request[3] addr 0x0A[3]				**BAR3 Burst Request**
Target Burst Request[2] addr 0x0A[2]				**BAR2 Burst Request**
Target Burst Request[1] addr 0x0A[1]				**BAR1 Burst Request**
Target Burst Request[0] addr 0x0A[0]				**BAR0 Burst Request** Prefetch enable on single dword/quadword request from PCI (only effective when Target Prefetch Control is set to "1") 1 Enable 0 Disable

Table 3-73
Control Register Offset 0x0C

Register Name: PCI_ID Register Offset: 0x0C			
Bits	**Function**		
63-56	0000_0000		
55-48	0000_0000		
47-40	0000_0000		
39-32	0000	I_2O Interrupt Mask Bit	000
31-24	0000_0000		
23-16	0000_0000		
15-8	0000_0000		
7-0	0000	I_2O Interrupt Service Request Bit	000

Table 3-74
I_2O Interrupt Mask Bit Description

Name	Type	Reset By	Reset Value	Function
I_2O Interrupt Mask Bit addr 0x0C[35]	Read/ Write (PCI)	TR/HTR	0	**I_2O Interrupt Status Register Mask** Set this bit to disable I_2O from generating a system interrupt.

Table 3-75
I_2O Interrupt Service Request Bit Description

Name	Type	Reset By	Reset Value	Function
I_2O Interrupt Status Request Bit addr 0x0C[3]	Read Only	TR/HTR	*0*	This bit is set to 1 if the I_2O Outbound Queue Pointer is an MFA. It is automatically set to 0 if the I_2O Outbound Queue Pointer is empty.

Table 3-76
Control Register Offset 0x0E

Register Name: PCI_ID Register Offset: 0x0E	
Bits	**Function**
63-56	*Reserved [63:56]*
55-48	*Reserved [55:48]*
47-40	*Reserved [47:40]*
39-32	*Reserved [39:32]*
31-24	*Reserved [31:24]*
23-16	*Reserved [23:16]*
15-8	*Reserved [15:8]*
7-0	*Reserved [7:0]*

Table 3-77
Reserved Register Description

Name	Type	Reset By	Reset Value	Function
Reserved[63:0] addr 0x0E[63:0]	N/A	N/A	*0*	Not Used. Must be "0".

Table 3-78
Control Register Offset 0x10

Register Name: PCI_ID Register Offset: 0x10	
Bits	**Function**
63-56	*I_2O Outbound Queue Pointer [31:24]*
55-48	*I_2O Outbound Queue Pointer [23:16]*
47-40	*I_2O Outbound Queue Pointer [15:8]*
39-32	*I_2O Outbound Queue Pointer [7:0]*
31-24	*I_2O Inbound Queue Pointer [31:24]*
23-16	*I_2O Inbound Queue Pointer [23:16]*
15-8	*I_2O Inbound Queue Pointer [15:8]*
7-0	*I_2O Inbound Queue Pointer [7:0]*

Table 3-79
I_2O Outbound Queue Pointer Description

Name	Type	Reset By	Reset Value	Function
I_2O Outbound Queue Pointer[31:0] addr 0x10[63:32]	Read (User) Write (PCI)	TR/HTR	0xFFFFFFFF	**I_2O Outbound Free List** This register overlaps with the I_2O Outbound Post List. User read returns Outbound Free List FIFO MFA. PCI Write places free MFA into the Outbound Free list FIFO. User read returns 0xFFFFFFFF when empty. Dword access only.
I_2O Outbound Queue Pointer[31:0] addr 0x10[63:32]	Read (PCI) Write (User)	TR/HTR	0xFFFFFFFF	**I_2O Outbound Post List** This register overlaps with the I_2O Outbound Free List. PCI read returns Outbound Post List FIFO MFA. User write places an MFA into the Outbound Post List FIFO. User read returns 0xFFFFFFFF when empty. Dword access only.

Table 3-80
I_2O Inbound Queue Pointer Description

Name	Type	Reset By	Reset Value	Function
I_2O Inbound Queue Pointer[31:0] addr 0x10[31:0]	Read (PCI) Write (User)	TR/HTR	0xFFFFFFFF	**I2O Inbound Free List** This register overlaps with the I_2O Inbound Post List. PCI read returns Inbound Free List FIFO MFA. User write places FREE MFA into the Inbound Free List FIFO. PCI read returns 0xFFFFFFFF when empty. Dword access only.
I_2O Inbound Queue Pointer[31:0] addr 0x10[31:0	Read (User) Write (PCI)	TR/HTR	0xFFFFFFFF	**I2O Inbound Post List** This register overlaps with the I_2O Inbound Free List. User read returns Inbound Post List FIFO MFA. PCI write places a MFA into the Outbound Post List FIFO. PCI read returns 0xFFFFFFFF when empty. Dword access only.

Table 3-81
Control Register Offset 0x12

Register Name: PCI_ID Register Offset: 0x12	
Bits	**Function**
63-56	*Master Read Address 0[63:56]/Chain Descriptor Start Address 0[63:56]*
55-48	*Master Read Address 0[55:48]/Chain Descriptor Start Address 0[55:48*
47-40	*Master Read Address 0[47:40]/Chain Descriptor Start Address 0[47:40]*
39-32	*Master Read Address 0[39:32]/Chain Descriptor Start Address 0[39:32]*
31-24	*Master Read Address 0[31:24]/Chain Descriptor Start Address 0[31:24]*
23-16	*Master Read Address 0[23:16]/Chain Descriptor Start Address 0[23:16]*
15-8	*Master Read Address 0[15:8]/Chain Descriptor Start Address 0[15:8]*
7-0	*Master Read Address 0[7:0]/Chain Descriptor Start Address 0[7:0]*

Table 3-82
Master Read Address0/Chain Descriptor Start Address Description

Name	Type	Reset By	Reset Value	Function
Master Read Address 0[63:0] addr 0x12[63:0] OR	Read/ Write	TR/HTR	0	**Starting Address for DMA Receive Channel 0** Establishes the starting PCI target address for data moving from the target to Receive FIFO0 under the control of Receive DMA Channel 0. The address may be any byte address.
Chain Descriptor Start Address 0[63:0] addr 0x12[63:0]	Read/ Write	TR/HTR	0	**FIrst Chain Descriptor Address** If chain mode—establishes the target PCI address for the first chain descriptor. Chain descriptors must be aligned on quadword boundaries. A simultaneous write from both the PCI bus and the ADSP-21160 cluster bus results in the PCI write being dropped.

Table 3-83
Control Register Offset 0x14

Register Name: PCI_ID Register Offset: 0x14	
Bits	**Function**
63-56	*Master Read Count Status 0 [31:24]*
55-48	*Master Read Count Status 0 [23:16]*
47-40	*Master Read Count Status 0 [15:8]*
39-32	*Master Read Count Status 0 [7:0]*
31-24	*Master Read Transfer Count 0 [31:24]*
23-16	*Master Read Transfer Count 0 [23:16]*
15-8	*Master Read Transfer Count 0 [15:8]*
7-0	*Master Read Transfer Count 0 [7:0]*

Table 3-84
Master Read Count Status 0 Description

Name	Type	Reset By	Reset Value	Function
Master Read Count Status 0[31:0] addr 0x14[63:32]	Read Only	TR/HTR	0	**Count Status for DMA Receive Channel 0** The status of the transfer operation for Receive DMA Channel 0 is monitored by reading this register. This register will contain the number of bytes yet to be transferred if Receive DMA Channel 0 is enabled. This register is accessible from both the PCI and ADSP-21160 cluster bus.

Table 3-85
Master Read Transfer Count 0 Description

Name	Type	Reset By	Reset Value	Function
Master Read Transfer Count 0[31:0] addr 0x14[31:0]	Read/ Write	TR/HTR	0	**Read Transfer Count for Receive Channel 0** Total number of bytes that are to be transferred from the PCI bus to Receive FIFO0 by Receive DMA Channel 0. Transfers may be any non-zero number up to 0xFFFFFFF0 (2^{32}–16). This register is read/write and is accessible from both the PCI and the ADSP-21160 cluster bus. A simultaneous write from both busses results in the PCI write being dropped.

Table 3-86
Control Register Offset 0x16

Register Name: PCI_ID Register Offset: 0x16	
Bits	**Function**
63-56	*Master Read Address 1[63:56]*
55-48	*Master Read Address 1[55:48]*
47-40	*Master Read Address 1[47:40]*
39-32	*Master Read Address 1[39:32]*
31-24	*Master Read Address 1[31:24]*
23-16	*Master Read Address 1[23:16]*
15-8	*Master Read Address 1[15:8]*
7-0	*Master Read Address 1[7:0]*

Table 3-87
Master Read Address 1 Description

Name	Type	Reset By	Reset Value	Function
Master Read Address 1[63:0] addr 0x16[63:0]	Read/ Write	TR/HTR	0	**Starting Address for DMA Receive Channel 1** Establishes the starting PCI target address for data moving from the target to Receive FIFO1 under the control of Receive DMA Channel 1. The address may be any byte address. A simultaneous write from both busses results in the PCI write being dropped.

Table 3-88
Control Register Offset 0x18

Register Name: PCI_ID Register Offset: 0x18	
Bits	**Function**
63-56	*Master Read Count Status 1 [31:24]*
55-48	*Master Read Count Status 1 [23:16]*
47-40	*Master Read Count Status 1 [15:8]*
39-32	*Master Read Count Status 1 [7:0]*
31-24	*Master Read Transfer Count 1 [31:24]*
23-16	*Master Read Transfer Count 1 [23:16]*
15-8	*Master Read Transfer Count 1 [15:8]*
7-0	*Master Read Transfer Count 1 [7:0]*

Table 3-89
Master Read Count Status 1 Description

Name	Type	Reset By	Reset Value	Function
Master Read Count Status 1[31:0] addr 0x18[63:32]	Read Only	TR/HTR	0	**Count Status for DMA Receive Channel 1** The status of the transfer operation for Receive DMA Channel 1 is monitored by reading this register. This register will contain the number of bytes yet to be transferred if Receive DMA Channel 1 is enabled. This register is accessible from both the PCI and the ADSP-21160 cluster bus.

Table 3-90
Master Read Transfer Count 1 Description

Name	Type	Reset By	Reset Value	Function
Master Read Transfer Count 1[31:0] addr 0x18[31:0]	Read/ Write	TR/HTR	*0*	**Read Transfer Count for Receive Channel 1** Total number of bytes that are to be transferred from the PCI bus to Receive FIFO1 by Receive DMA Channel 1. Transfers may be any non-zero number up to 0xFFFFFFF0 (2^{32}–16). This register is read/write and is accessible from both the PCI and the ADSP-21160 cluster bus. A simultaneous write from both busses results in the PCI write being dropped.

Table 3-91
Control Register Offset 0x1A

Register Name: PCI_ID Register Offset: 0x1A		
Bits	**Reset By**	**Function**
63-56	00	*XMIT FIFO1 Almost Full[5:0]*
55-48	00	*XMIT FIFO0 Almost Full[5:0]*
47-40	00	*XMIT FIFO1 Almost Empty[5:0]*
39-32	00	*XMIT FIFO0 Almost Empty[5:0]*
31-24	00	*Receive FIFO1 Almost Full[5:0]*
23-16	00	*Receive FIFO0 Almost Full[5:0]*
15-8	00	*Receive FIFO1 Almost Empty[5:0]*
7-0	00	*Receive FIFO0 Almost Full[5:0]*

Table 3-92
XMIT FIFO Flags Description

Name	Type	Reset By	Reset Value	Function
XMIT FIFO1 Almost Full[5:0] addr 0x1A[63:56]	Read/ Write	TR/ HTR	08	**Transmit FIFO1 Programmable Almost Full Flag** Almost full flag will go active when total number of quadwords that may be written to Transmit FIFO1 is equal to or less than the number set in this register. Controls when the SHARC interface DMA engine is told to hold off a burst.
XMIT FIFO0 Almost Full[5:0] addr 0x1A[55:48]	Read/ Write	TR/ HTR	08	**Transmit FIFO0 Programmable Almost Full Flag** Almost full flag will go active when total number of quadwords that may be written to Transmit FIFO0 is equal to or less than the number set in this register. Controls when the SHARC interface DMA engine is told to hold off a burst.
XMIT FIFO1 Almost Empty[5:0] addr 0x1A[47:40]	Read/ Write	TR/ HTR	08	**Transmit FIFO1 Programmable Almost Empty Flag** Almost empty flag will go active when total number of quadwords that are available in Transmit FIFO1 is equal to or greater than the number set in this register. Used to guarantee a minimum number of bursts during master transmit operations.
XMIT FIFO0 Almost Empty[5:0] addr 0x1A[39:32]	Read/ Write	TR/ HTR	08	**Transmit FIFO0 Programmable Almost Empty Flag** Almost empty flag will go active when total number of quadwords that are available in Transmit FIFO0 is equal to or greater than the number set in this register. Used to guarantee a minimum number of bursts during master transmit operations.

Table 3-93
Receive FIFO Flags Description

Name	Type	Reset By	Reset Value	Function
Receive FIFO1 Almost Full[5:0] addr 0x1A[31:24]	Read/ Write	TR/ HTR	08	**Receive FIFO1 Programmable Almost Full Flag** Almost full flag will go active when total number of quadwords that may be written to Receive FIFO1 is equal to or less than the number set in this register. Used to guarantee a minimum number of bursts during master receive operations.
Receive FIFO0 Almost Full[5:0] addr 0x1A[23:16]	Read/ Write	TR/ HTR	08	**Receive FIFO0 Programmable Almost Full Flag** Almost full flag will go active when total number of quadwords that may be written to Receive FIFO0 is equal to or less than the number set in this register. Used to guarantee a minimum number of bursts during master receive operations.
Receive FIFO1 Almost Empty[5:0] addr 0x1A[15:8]	Read/ Write	TR/ HTR	08	**Receive FIFO1 Programmable Almost Empty Flag** Almost empty flag will go active when total number of quadwords that are available in Receive FIFO1 is equal to or greater than the number set in this register. Controls when the SHARC interface DMA engine is allowed to burst.
Receive FIFO0 Almost Empty[5:0] addr 0x1A[7:0]	Read/ Write	TR/ HTR	08	**Receive FIFO0 Programmable Almost Empty Flag** Almost empty flag will go active when total number of quadwords that are available in Receive FIFO0 is equal to or greater than the number set in this register. Controls when the SHARC interface DMA engine is allowed to burst.

Table 3-94
Control Register Offset 0x1C

Register Name: PCI_ID Register Offset: 0x1C	
Bits	**Function**
63-56	*PCI Outgoing Mail 7 [7:0]/User Incoming Mail 7 [7:0]*
55-48	*PCI Outgoing Mail 6 [7:0]/User Incoming Mail 6 [7:0]*
47-40	*PCI Outgoing Mail 5 [7:0]/User Incoming Mail 5 [7:0]*
39-32	*PCI Outgoing Mail 4 [7:0]/User Incoming Mail 4 [7:0]*
31-24	*PCI Outgoing Mail 3 [7:0]/User Incoming Mail 3 [7:0]*
23-16	*PCI Outgoing Mail 2 [7:0]/User Incoming Mail 2 [7:0]*
15-8	*PCI Outgoing Mail 1 [7:0]/User Incoming Mail 1 [7:0]*
7-0	*PCI Outgoing Mail 0 [7:0]/User Incoming Mail 0 [7:0]*

Table 3-95
PCI Outgoing Mail Description

Name	Type	Reset By	Reset Value	Function
PCI Outgoing Mail 7 [7:0] *User Incoming Mail 7 [7:0]* addr 0x1C[63:56]	Write/PCI Read/User	TR/ HTR	*0*	**PCI Outgoing Mailbox 7** Mailbox data written by PCI to be read via the ADSP-21160 cluster bus. On a PCI write—sets the *PCI Outgoing Mailbox Status [7]/User Incoming Mailbox Status [7]* and the *User Incoming Mailbox Full Interrupt Flag [7].* On a ADSP-21160 cluster bus read—clears the *PCI Outgoing Mailbox Status [7] / User Incoming Mailbox Status [7]* and sets the *PCI Outgoing Mailbox Empty Interrupt Flag [7].*
(Sheet 1 of 4)				

Table 3-95
PCI Outgoing Mail Description (Continued)

Name	Type	Reset By	Reset Value	Function
PCI Outgoing Mail 6 [7:0] *User Incoming Mail 6[7:0]* addr 0x1C[55:48]	Write/PCI Read/User	TR/ HTR	0	**PCI Outgoing Mailbox 6** Mailbox data written by PCI to be read via the ADSP-21160 cluster bus. On a PCI write—sets the *PCI Outgoing Mailbox Status [6]/User Incoming Mailbox Status [6]* and the *User Incoming Mailbox Full Interrupt Flag [6].* On an ADSP-21160 cluster bus read—clears the *PCI Outgoing Mailbox Status [6] / User Incoming Mailbox Status [6]* and sets the PCI *Outgoing Mailbox Empty Interrupt Flag [6].*
PCI Outgoing Mail 5 [7:0] U*ser Incoming Mail 5[7:0]* **a**ddr 0x1C[47:40]	Write/PCI Read/User	TR/ HTR	0	**PCI Outgoing Mailbox 5** Mailbox data written by PCI to be read via the ADSP-21160 cluster bus. On a PCI write—sets the *PCI Outgoing Mailbox Status [5]/User Incoming Mailbox Status [5]* and the *User Incoming Mailbox Full Interrupt Flag [5].* On a ADSP-21160 cluster bus read—clears the *PCI Outgoing Mailbox Status [5] / User Incoming Mailbox Status [5]* and sets the *PCI Outgoing Mailbox Empty Interrupt Flag [5].*
PCI Outgoing Mail 4 [7:0] *User Incoming Mail 4[7:0]* addr 0x1C[39:32]	Write/PCI Read/User	TR/ HTR	0	**PCI Outgoing Mailbox 4** Mailbox data written by PCI to be read via the ADSP-21160 cluster bus. On a PCI write—sets the *PCI Outgoing MB Status [4]/User Incoming MB Status [4]* and the *User Incoming MB Full Interrupt Flag [4].* On a ADSP-21160 cluster bus read—clears the *PCI Outgoing MB Status [4] / User Incoming MB Status [4]* and sets the *PCI Outgoing MB Empty Interrupt Flag [4].*
(Sheet 2 of 4)				

Table 3-95
PCI Outgoing Mail Description (Continued)

Name	Type	Reset By	Reset Value	Function
PCI Outgoing Mail 3 [7:0] *User Incoming Mail 3[7:0]* addr 0x1C[31:24]	Write/PCI Read/User	TR/ HTR	0	**PCI Outgoing Mailbox 3** Mailbox data written by PCI to be read via the ADSP-21160 cluster bus. On a PCI write—sets the *PCI Outgoing Mailbox Status [3]/User Incoming Mailbox Status [3]* and the *User Incoming Mailbox Full Interrupt Flag [3].* On a ADSP-21160 cluster bus read—clears the *PCI Outgoing Mailbox Status [3] / User Incoming Mailbox Status [3]* and sets the P*CI Outgoing Mailbox Empty Interrupt Flag [3].*
PCI Outgoing Mail 2 [7:0] *User Incoming Mail 2[7:0]* addr 0x1C[23:16]	Write/PCI Read/User	TR/ HTR	0	**PCI Outgoing Mailbox 2** Mailbox data written by PCI to be read via the ADSP-21160 cluster bus. On a PCI write—sets the *PCI Outgoing Mailbox Status [2]/User Incoming Mailbox Status [2]* and the *User Incoming Mailbox Full Interrupt Flag [2].* On an ADSP-21160 cluster bus read—clears the *PCI Outgoing Mailbox Status [2] / User Incoming Mailbox Status [2]* and sets the *PCI Outgoing Mailbox Empty Interrupt Flag [2].*
(Sheet 3 of 4)				

Table 3-95
PCI Outgoing Mail Description (Continued)

Name	Type	Reset By	Reset Value	Function
PCI Outgoing Mail 1 [7:0] *User Incoming Mail 1[7:0]* addr 0x1C[15:8]	Write/PCI Read/User	TR/ HTR	0	**PCI Outgoing Mailbox 1** Mailbox data written by PCI to be read via the ADSP-21160 cluster bus. On a PCI write—sets the *PCI Outgoing Mailbox Status [1]/User Incoming Mailbox Status [1]* and the U*ser Incoming Mailbox Full Interrupt Flag [1].* On a ADSP-21160 cluster bus read—clears the *PCI Outgoing Mailbox Status [1] / User Incoming Mailbox Status [1]* and sets the *PCI Outgoing Mailbox Empty Interrupt Flag [1].*
PCI Outgoing Mail 0 [7:0] *User Incoming Mail 0[7:0]* addr 0x1C[7:0]	Write/PCI Read/User	TR/ HTR	0	**PCI Outgoing Mailbox 0** Mailbox data written by PCI to be read via the ADSP-21160 cluster bus. On a PCI write—sets the *PCI Outgoing Mailbox Status [0]/User Incoming Mailbox Status [0]* and the *User Incoming Mailbox Full Interrupt Flag [0].* On a ADSP-21160 cluster bus read—clears the *PCI Outgoing Mailbox Status [0] / User Incoming Mailbox Status [0]* and sets the *PCI Outgoing Mailbox Empty Interrupt Flag [0].*
(Sheet 4 of 4)				

Table 3-96
Control Register Offset 0x1E

Register Name: PCI_ID Register Offset: 0x1E	
Bits	**Function**
63-56	*PCI Incoming Mail 7 [7:0]* *User Outgoing Mail 7 [7:0]*
55-48	*PCI Incoming Mail 6 [7:0]* *User Outgoing Mail 6 [7:0]*
47-40	*PCI Incoming Mail 5 [7:0]* *User Outgoing Mail 5 [7:0]*
39-32	*PCI Incoming Mail 4 [7:0]* *User Outgoing Mail 4 [7:0]*
31-24	*PCI Incoming Mail 3 [7:0]* *User Outgoing Mail 3 [7:0]*
23-16	*PCI Incoming Mail 2 [7:0]* *User Outgoing Mail 2 [7:0]*
15-8	*PCI Incoming Mail 1 [7:0]* *User Outgoing Mail 1 [7:0]*
7-0	*PCI Incoming Mail 0 [7:0]* *User Outgoing Mail 0 [7:0]*

Table 3-97
PCI Incoming Mail Description

Name	Type	Reset By	Reset Value	Function
User Outgoing Mail 7 [7:0] *PCI Incoming Mail 7 [7:0]* addr 0x1E[63:56]	Write/User Read/PCI	TR/ HTR	*0*	**User Outgoing Mailbox 7** Mailbox data written by ADSP-21160 cluster bus to be read via PCI. On an ADSP-21160 cluster bus write—sets the *User Outgoing Mailbox Status [7] / PCI Incoming Mailbox Status [7]* and the *PCI Incoming Mailbox Full Interrupt Flag [7].* On a PCI read—clears the *User Outgoing Mailbox Status [7] / PCI Incoming Mailbox Status [7]* and sets the *User Outgoing Mailbox Empty Interrupt Flag [7].*
User Outgoing Mail 6 [7:0] *PCI Incoming Mail 6 [7:0]* addr 0x1E[55:48]	Write/User Read/PCI	TR/ HTR	0	**User Outgoing Mailbox 6** Mailbox data written by ADSP-21160 cluster bus to be read via PCI. On an ADSP-21160 cluster bus write—sets the *User Outgoing Mailbox Status [6] / PCI Incoming Mailbox Status [6]* and the *PCI Incoming Mailbox Full Interrupt Flag [6].* On a PCI read—clears the *User Outgoing Mailbox Status [6] / PCI Incoming Mailbox Status [6]* and sets the *User Outgoing Mailbox Empty Interrupt Flag [6].*
User Outgoing Mail 5 [7:0] *PCI Incoming Mail 5 [7:0]* addr 0x1E[47:40]	Write/User Read/PCI	TR/ HTR	0	**User Outgoing Mailbox 5** Mailbox data written by ADSP-21160 cluster bus to be read via PCI. On an ADSP-21160 cluster bus write—sets the *User Outgoing Mailbox Status [5] / PCI Incoming Mailbox Status [5]* and the *PCI Incoming Mailbox Full Interrupt Flag [5].* On a PCI read—clears the *User Outgoing Mailbox Status [5] / PCI Incoming Mailbox Status [5]* and sets the *User Outgoing Mailbox Empty Interrupt Flag [5].*
(Sheet 1 of 4)				

Table 3-97
PCI Incoming Mail Description (Continued)

Name	Type	Reset By	Reset Value	Function
User Outgoing Mail 4 [7:0] *PCI Incoming Mail 4 [7:0]* addr 0x1E[39:32]	Write/User Read/PCI	TR/ HTR	0	**User Outgoing Mailbox 4** Mailbox data written by ADSP-21160 cluster bus to be read via PCI. On an ADSP-21160 cluster bus write—sets the *User Outgoing Mailbox Status [4] / PCI Incoming Mailbox Status [4]* and the *PCI Incoming Mailbox Full Interrupt Flag [4].* On a PCI read—clears the *User Outgoing Mailbox Status [4] / PCI Incoming Mailbox Status [4]* and sets the *User Outgoing Mailbox Empty Interrupt Flag [4].*
User Outgoing Mail 3 [7:0] *PCI Incoming Mail 3 [7:0]* addr 0x1E[31:24]	Write/User Read/PCI	TR/ HTR	0	**User Outgoing Mailbox 3** Mailbox data written by ADSP-21160 cluster bus to be read via PCI. On an ADSP-21160 cluster bus write—sets the *User Outgoing Mailbox Status [3] / PCI Incoming Mailbox Status [3]* and the PCI Incoming Mailbox Full Interrupt Flag [3]. On a PCI read—clears the *User Outgoing Mailbox Status [3] / PCI Incoming Mailbox Status [3]* and sets the *User Outgoing Mailbox Empty Interrupt Flag [3].*

(Sheet 2 of 4)

Table 3-97
PCI Incoming Mail Description (Continued)

Name	Type	Reset By	Reset Value	Function
User Outgoing Mail 2 [7:0] *PCI Incoming Mail 2 [7:0]* addr 0x1E[23:16]	Write/User Read/PCI	TR/ HTR	0	**User Outgoing Mailbox 2** Mailbox data written by ADSP-21160 cluster bus to be read via PCI. On an ADSP-21160 cluster bus write—sets the *User Outgoing Mailbox Status [2] / PCI Incoming Mailbox Status [2]* and the *PCI Incoming Mailbox Full Interrupt Flag [2].* On a PCI read—clears the *User Outgoing Mailbox Status [2] / PCI Incoming Mailbox Status [2]* and sets the *User Outgoing Mailbox Empty Interrupt Flag [2].*
(Sheet 3 of 4)				

Table 3-97
PCI Incoming Mail Description (Continued)

Name	Type	Reset By	Reset Value	Function
User Outgoing Mail 1 [7:0] *PCI Incoming Mail 1 [7:0]* addr 0x1E[15:8]	Write/User Read/PCI	TR/ HTR	0	**User Outgoing Mailbox 1** Mailbox data written by ADSP-21160 cluster bus to be read via PCI. On an ADSP-21160 cluster bus write—sets the *User Outgoing Mailbox Status [1] / PCI Incoming Mailbox Status [1]* and the *PCI Incoming Mailbox Full Interrupt Flag [1].* On a PCI read—clears the *User Outgoing Mailbox Status [1] / PCI Incoming Mailbox Status [1]* and sets the *User Outgoing Mailbox Empty Interrupt Flag [1].*
User Outgoing Mail 0 [7:0] *PCI Incoming Mail 0 [7:0]* addr 0x1E[7:0]	Write/User Read/PCI	TR/ HTR	0	**User Outgoing Mailbox 0** Mailbox data written by ADSP-21160 cluster bus to be read via PCI. On an ADSP-21160 cluster bus write—sets the *User Outgoing Mailbox Status [0] / PCI Incoming Mailbox Status [0]* and the *PCI Incoming Mailbox Full Interrupt Flag [0].* On a PCI read—clears the *User Outgoing Mailbox Status [0] / PCI Incoming Mailbox Status [0]* and sets the *User Outgoing Mailbox Empty Interrupt Flag [0].*
(Sheet 4 of 4)				

Table 3-98
Control Register Offset 0x20

Register Name: PCI_ID Register Offset: 0x20				
Bits	**Function**			
63-56	*0000_000*			*User Int [PCI Only]*
55-48	*00*	*DMA Interrupt [5:0] [PCI Only]*		
47-40	*PCI Incoming MB Full [7:0] [PCI Only]*			
39-32	*PCI Outgoing MB Empty [7:0] [PCI Only]*			
31-24	*0000*		*I_2O Interrupt[3:0] [User Only]*	
23-16	*BIST Start Interrupt [User]*	*Single PCI Interrupt [User]*	*DMA Interrupt[5:0] [User Only]*	
15-8	*User Outgoing MB Empty[7:0] [User Only]*			
7-0	*User Incoming MB Full[7:0] [User Only]*			

Note:
For addr 0x20 and 0x22, all registers labeled as "user" are used for the generation of **PCFlg** (should match interface signal name).
All registers labeled "PCI" are used for the generation of **INTA#** (should match pin name).

Table 3-99
User Interrupt Description

Name	Type	Reset By	Reset Value	Function
User Interrupt addr 0x20[56]	Read Only	TR/ HTR	*0*	User Interrupt (PCI Only) Status of the signal **PCInt**: 1 **PCInt**==1 0 **PCInt**==0

Table 3-100
DMA Interrupt Description

Name	Type	Reset By	Reset Value	Function
DMA Interrupt[5] addr 0x20[53] PCI addr 0x20[21] user	Read/Write 1 to Clear	TR/ HTR	*0*	**Chain Descriptor Fetch End Interrupt** 1 Last Chain Descriptor has been fetched (EOC detected) 0 EOC has not been encountered
DMA Interrupt[4] addr 0x20[52] PCI addr 0x20[22] user	Read/Write 1 to Clear	TR/ HTR	0	**Chain Mode DMA Interrupt** 1 Chain Mode has completed all data transfers 0 Chain Mode is idle or has not completed all data transfers
DMA Interrupt[3] addr 0x20[51] PCI addr 0x20[19] user	Read/Write 1 to Clear	TR/ HTR	0	**Receive DMA Channel 1 Interrupt** 1 Receive DMA Channel 1 has completed all requested data transfers. 0 Receive DMA Channel 1 has not completed all requested data transfers.
DMA Interrupt[2] addr 0x20[50] PCI addr 0x20[18] user	Read/Write 1 to Clear	TR/ HTR	0	**Receive DMA Channel 0 Interrupt** 1 Receive DMA Channel 0 has completed all requested data transfers. 0 Receive DMA Channel 0 has not completed all requested data transfers.
DMA Interrupt[1] addr 0x20[49] PCI addr 0x20[17] user	Read/Write 1 to Clear	TR/ HTR	0	**Transmit DMA Channel 1 Interrupt** 1 Transmit DMA Channel 1 has completed all requested data transfers. 0 Transmit DMA Channel 1 has not completed all requested data transfers.
DMA Interrupt [0] addr 0x20[48] PCI addr 0x20[16] user	Read/Write 1 to Clear	TR/ HTR	0	**Transmit DMA Channel 0 Interrupt** 1 Transmit DMA Channel 0 has completed all requested data transfers. 0 Transmit DMA Channel 0 has not completed all requested data transfers.

Table 3-101
PCI Incoming MB Full Description

Name	Type	Reset By	Reset Value	Function
PCI Incoming Mail Full [7:0] Status addr 0x20[47:40]	Read/ Write 1 to Clear	TR/ HTR	0	**PCI Incoming Mail Full Status[7:0]** 1 PCI Incoming Mailbox is full 0 PCI Incoming Mailbox is not full

Table 3-102
PCI Outgoing MB Empty

Name	Type	Reset By	Reset Value	Function
PCI Outgoing Mail Empty [7:0] Status addr 0x20[39:32]	Read/ Write 1 to Clear	TR/ HTR	0	**PCI Outgoing Mail Empty Status[7:0]** 1 PCI Outgoing Mailbox is empty 0 PCI Outgoing Mailbox is not empty

Table 3-103
I_2O Interrupt Description

Name	Type	Reset By	Reset Value	Function
I_2O Interrupt[3] addr 0x20[27]	Read /Write 1 to Clear	TR/HTR	1	**Outbound Free List Full** 1 PCI writes to I_2O Outbound Queue Pointer 0 Write 1 to clear
I_2O Interrupt[2] addr 0x20[26]	Read /Write 1 to Clear	TR/HTR	0	**Outbound Post List Empty** 1 PCI reads from I_2O Outbound Queue Pointer 0 Write 1 to clear
I_2O Interrupt[1] addr 0x20[25]	Read /Write 1 to Clear	TR/HTR	0	**Inbound Post List Full** 1 PCI writes to I_2O Outbound Queue Pointer 0 Write 1 to clear
I_2O Interrupt[0] addr 0x20[24]	Read /Write 1 to Clear	TR/HTR	1	**Inbound Free List Empty** 1 PCI reads from I_2O Outbound Queue Pointer 0 Write 1 to clear

Table 3-104
BIST Start Interrupt Description

Name	Type	Reset By	Reset Value	Function
BIST Start Interrupt addr 0x20[23]	Read /Write 1 to Clear	TR/ HTR	0	**Built In Self Test (BIST) Interrupt** 1 Built In Self Test (BIST) Interrupt Requested 0 No BIST Interrupt pending

Table 3-105
SPCI Interrupt Description

Name	Type	Reset By	Reset Value	Function
Single PCI Interrupt addr 0x20[22]	Read /Write 1 to Clear	TR/ HTR	0	**Single PCI Interrupt** 1 Single PCI Controller has completed 0 No SPCI interrupt pending May be accessed from ADSP-21160 cluster bus only

Table 3-106
User Outgoing MB Empty Description

Name	Type	Reset By	Reset Value	Function
User Output Mail Empty Interrupt Status [7:0] addr 0x20[15:8]	Read /Write 1 to Clear	TR/ HTR	0	**User Output Mail Empty Interrupt Status** 1 Interrupt pending 0 No interrupt pending May be accessed from ADSP-21160 cluster bus only

Table 3-107
User Incoming MB Full Description

Name	Type	Reset By	Reset Value	Function
PCI Output Mail Full Interrupt Status [7:0] addr 0x20[7:0]	Read /Write 1 to Clear	TR/ HTR	0	**PCI Output Mail Full Interrupt Status [7:0]** 1 Interrupt pending 0 No interrupt pending May be accessed from ADSP-21160 cluster bus only

Table 3-108
Control Register Offset 0x22

Register Name: PCI_ID Register Offset: 0x22				
Bits	**Function**			
63-56	*0000_000*			*User Mask [PCI]*
55-48	*00*	*DMA Interrupt Mask[5:0] [PCI Only]*		
47-40	*PCI Incoming Mail Box Full Interrupt Mask[7:0]/ User OMB Full Interrupt Mask[7:0] (PCI only)*			
39-32	*PCI OMB Empty Interrupt Mask[7:0]/ User Incoming Mail Box Empty Interrupt Mask[7:0] (PCI Only)*			
31-24	*0000*		*I_2O Interrupt Mask[3:0] [User Only]*	
23-16	*BIST Mask [User]*	*Single PCI Interrupt Mask [User]*	*DMA Interrupt Mask[5:0] [User Only]*	
15-8	*PCI Incoming Mail Box Empty interrupt Mask[7:0]/ User OMB Empty Interrupt Mask[7:0] (user Only)*			
7-0	*PCI OMB Full Interrupt Mask[7:0]/ User Incoming Mailbox Full Interrupt Mask[7:0] (user Only)*			

Note:
For addr 0x20 and 0x22, all registers labeled as "user" are used for the generation of **PCFlg** (should match interface signal name).
All registers labeled "PCI" are used for the generation of **$\overline{\text{INTA}}$** (should match pin name).

Table 3-109
User Interrupt Mask Description

Name	Type	Reset By	Reset Value	Function
User interrupt Mask addr 0x22[56]	Read/Write	TR/ HTR	0	**User Interrupt Mask** 1 Allow interrupts on **PCInt** 0 Mask interrupts on **PCInt**

Table 3-110
DMA Interrupt Mask Description

Name	Type	Reset By	Reset Value	Function
DMA Interrupt Mask[5] addr 0x22[53] PCI addr 0x22[21] user	Read/Write	TR/ HTR	0	**Chain Descriptor Fetch End Interrupt** 1 Allow interrupt 0 Mask interrupt
DMA Interrupt Mask[4] addr 0x22[52] PCI addr 0x22[20] user	Read/Write	TR/ HTR	0	**Chain Mode DMA Interrupt** 1 Allow interrupt 0 Mask interrupt
DMA Interrupt Mask[3] addr 0x22[51] PCI addr 0x22[19] user	Read/Write	TR/ HTR	0	**Receive Channel 1 Interrupt** 1 Allow interrupt 0 Mask interrupt
DMA Interrupt Mask[2] addr 0x22[50] PCI addr 0x22[18] user	Read/Write	TR/ HTR	0	**Receive Channel 0 Interrupt** 1 Allow interrupt 0 Mask interrupt
DMA Interrupt Mask[1] addr 0x22[49] PCI addr 0x22[17] user	Read/Write	TR/ HTR	0	**Transmit Channel 1 Interrupt** 1 Allow interrupt 0 Mask interrupt
DMA Interrupt Mask[0] addr 0x22[48] PCI addr 0x22[16] user	Read/Write	TR/ HTR	0	**Transmit Channel 0 Interrupt** 1 Allow interrupt 0 Mask interrupt

Table 3-111
PCI Incoming Mailbox Full Interrupt Mask/User OMB Full Interrupt Mask

Name	Type	Reset By	Reset Value	Function
PCI IMB Full Interrupt Mask [7:0] *User OMB Full Interrupt Mask [7:0]* addr 0x22[47:40]	Read/ Write	TR/HTR	0	**PCI Incoming Mailbox Full Interrupt Mask/User Outgoing Mailbox Full Interrupt Mask** 1 Enable Interrupt 0 Disable Interrupt

Table 3-112
PCI OMB Empty Interrupt Mask/User IMB Empty Interrupt Mask

Name	Type	Reset By	Reset Value	Function
PCI OMB Empty Interrupt Mask [7:0] *User IMB Empty Interrupt Mask [7:0]* addr 0x22[39:32]	Read/ Write	TR/HTR	*0*	**PCI Outgoing Mailbox Empty Interrupt Mask/User Incoming Mailbox Empty Interrupt Mask** 1 Enable Interrupt 0 Disable Interrupt

Table 3-113
I_2O Interrupt Mask

Name	Type	Reset By	Reset Value	Function
I_2O Interrupt Mask[3] addr 0x22[27]	Read/Write	TR/HTR	0	**Outbound Free List Full Interrupt Mask** 1 Enable Interrupt 0 Disable Interrupt
I_2O Interrupt Mask[2] addr 0x22[26]	Read/Write	TR/HTR	0	**Outbound Post List Empty Interrupt Mask** 1 Enable Interrupt 0 Disable Interrupt
I2O Interrupt Mask[1] addr 0x22[25]	Read/Write	TR/HTR	0	**Inbound Post List Full Interrupt Mask** 1 Enable Interrupt 0 Disable Interrupt
I_2O Interrupt Mask[0] addr 0x22[24]	Read/Write	TR/HTR	0	**Inbound Free List Empty Interrupt Mask** 1 Enable Interrupt 0 Disable Interrupt

Table 3-114
BIST Mask

Name	Type	Reset By	Reset Value	Function
BIST Mask addr 0x22[23]	Read/Write	TR/HTR	0	**Built-In Self Test Interrupt Mask** 1 Enable Interrupt 0 Disable Interrupt

Table 3-115
Single PCI (SPCI) Interrupt Mask

Name	Type	Reset By	Reset Value	Function
Single PCI Interrupt Mask addr 0x22[22]	Read/Write	TR/ HTR	0	**Single PCI Interrupt Mask** 1 Enable Interrupt 0 Disable Interrupt

Table 3-116
PCI IMB Empty Interrupt Mask/User OMB Empty Interrupt Mask

Name	Type	Reset By	Reset Value	Function
PCI IMB Empty Interrupt Mask[7:0] *User OMB Empty Interrupt Mask[7:0]* addr 0x22[15:8]	Read/Write	TR/ HTR	0	**PCI IMB Empty Interrupt Mask [7:0]** **User OMB Empty Interrupt Mask [7:0]** 1 Enable Interrupt 0 Disable Interrupt

Table 3-117
PCI OMB Full Interrupt Mask/User IMB Full Interrupt Mask

Name	Type	Reset By	Reset Value	Function
PCI OMB Full Interrupt Mask [7:0] *User IMB Full Interrupt Mask [7:0]* addr 0x22[7:0]	Read/Write	TR/ HTR	0	**PCI OMB Full Interrupt Mask [7:0]** **User IMB Full Interrupt Mask [7:0]** 1 Enable Interrupt 0 Disable Interrupt

Table 3-118
Control Register Offset 0x24

Register Name: PCI_ID Register Offset: 0x24						
Bits	**Function**					
63-56	*0000*				*PCI Error Status[3:0] [PED, MRT, MA. RTA]*	
55-48	*00*		*Master Channel Error Status[5:0]*			
47-40	*0*	*Target User RDWR [User]*	*Target Address Valid [User]*	*Target User Request [User]*	*Target User Multiple [User]*	*barCS[2:0] [User Only]*
39-32	*user_be_req[7:0] [User Only]*					
31-24	*0000*				*I_2O Status[3:0]*	
23-16	*BIST Start [User]*	*0*	*Chain Pointer Fetch End*	*DMA Start/Done#[4:0]*		
15-8	*PCI Incoming MB Status[7:0]/User Outgoing MB Status[7:0]*					
7-0	*PCI Outgoing MB Status[7:0]/User Incoming MB Status[7:0]*					

Table 3-119
PCI Error Status Flags Description

Name	Type	Reset By	Reset Value	Function
PCI Error Status[3] Parity Error Flag Detected (PED) addr 0x24[42:40]	Read /Write 1 to Clear	TR/ HTR	0	**Parity Error** 1 Parity Error Detected 0 No parity errors detected
PCI Error Status[2] Maximum Retry Flag Timeout (MTA) addr 0x24[58]	Read /Write 1 to Clear	TR/ HTR	0	**Maximum Retry Timeout** 1 Retry Timeout 0 No Retry Timeout
PCI Error Status[1] Master Abort (MA) addr 0x24[57])	Read /Write 1 to Clear	TR/ HTR	0	**Master Abort** 1 Master Abort has occurred 0 No master aborts have occurred
PCI Error Status [0] Received a Target Abort addr 0x24[56]	Read /Write 1 to Clear	TR/ HTR	0	**Received a Target Abort** 1 Target abort has occurred 0 No target aborts have occurred

Table 3-120
Error Description

Name	Type	Reset By	Reset Value	Function
Master Channel Error Status[5] Single PCI Access (SPCI) addr 0x24[53]	Read Only	TR/ HTR	0	**SPCI Error** 1 An error has occurred on a SPCI access 0 No errors
Master Channel Error Status[4] Chain DMA Mode addr 0x24[52]	Read Only	TR/ HTR	0	**Chain Mode Error** 1 An error has occurred during a Chain Mode access 0 No errors
Master Channel Error Status[3] Receive DMA Channel 1 addr 0x24[51]	Read Only	TR/ HTR	0	**Receive DMA Channel 1** 1 An error has occurred during a Receive DMA Channel 1 access 0 No errors
Master Channel Error Status[2] Receive DMA Channel 0 addr 0x24[50]	Read Only	TR/ HTR	0	**Receive DMA Channel 0** 1 An error has occurred during a Receive DMA Channel 0 access 0 No errors
Master Channel Error Status[1] Transmit DMA Channel 1 addr 0x24[49]	Read Only	TR/ HTR	0	**Transmit DMA Channel 1** 1 An error has occurred during a Transmit DMA Channel 1 access 0 No errors

(Sheet 1 of 2)

Table 3-120
Error Description (Continued)

Name	Type	Reset By	Reset Value	Function
Master Channel Error Status[0] Transmit DMA Channel 0 addr 0x24[48]	Read Only	TR/ HTR	0	**Transmit DMA Channel 0** 1 An error has occurred during a Transmit DMA Channel 0 access 0 No errors

(Sheet 2 of 2)

Table 3-121
Target Interface Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x24[46]	Read Only	TR/ HTR	0	**Reserved**

Table 3-122
Target Address Valid Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x24[45]	Read Only	TR/ HTR	0	**Reserved**

Table 3-123
Target User Request Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x24[44]	Read Only	TR/ HTR	0	**Reserved**

Table 3-124
Target User Multiple Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x24[43]	Read Only	TR/ HTR	0	**Reserved**

Table 3-125
barCS Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x24[42:40]	Read Only	TR/ HTR	0	**Reserved**

target_region	Function (32 bits)	Function (64 bits)
000	Base Address Region 0 Match (≥0x100 only)	Base Address Region 0 Match (≥0x100 only)
001	Base Address Region 1 Match	Not Applicable
010	Base Address Region 2 Match	Base Address Region 2 Match
011	Base Address Region 3 Match	Not Applicable
100	Base Address Region 4 Match	Base Address Region 4 Match
101	Base Address Region 5 Match	Not Applicable
110	Expansion ROM	Expansion ROM (32 bits only)
111	Configuration Space (0x40–0x7FF)	Configuration Space (0x40–0x7FF) (32 bits only)

Table 3-126
user_be_req Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x24[39:32]	Read Only	TR/ HTR	0	**Reserved**

Table 3-127
I_2O Status Description

Name	Type	Reset By	Reset Value	Function
I_2O Status[3] addr 0x24[27]	Read Only	TR/HTR	0	**I_2O Outbound Free List Full** 1 PCI has written to the I_2O Outbound Queue Pointer returning a free frame pointer to the Outbound Free List Register. The register is full. 0 ADSP-21160 cluster bus has read from the Outbound Queue Pointer, fetching a free frame pointer from the Outbound Free List Register. The register is empty.
I_2O Status[2] addr 0x24[26]	Read Only	TR/HTR	1	**I_2O Outbound Post List Empty** 1 PCI has read from the I_2O Outbound Queue Pointer, fetching a message frame pointer from the Outbound Post List Register. The register is empty. 0 ADSP-21160 cluster bus has written to the I_2O Outbound Queue Pointer, placing a message frame pointer in the Outbound Post List Register. The register is full.
I_2O Status[1] addr 0x24[25]	Read Only	TR/HTR	0	**I_2O Inbound Post List Full** 1 PCI has written to the inbound queue pointer, placing a message frame pointer in the Inbound Post List Register. The register is full. 0 ADSP-21160 cluster bus has read from the inbound queue pointer, fetching a message frame pointer from the Inbound Post List Register. The register is empty.
I_2O Status[0] addr 0x24[24]	Read Only	TR/HTR	1	**I_2O Inbound Free List Empty** Inbound Free List Register is empty. 1 PCI has read from the inbound queue pointer, fetching a free frame pointer from the Inbound Free List Register. The register is empty. 0 ADSP-21160 cluster bus has written to the inbound queue pointer, returning a free frame pointer to the Inbound Free List Register. The register is full.

Table 3-128
BIST Start Description

Name	Type	Reset By	Reset Value	Function
BIST Start addr 0x24[23]	Read Only	TR/ HTR	0	**BIST Status** 1 PCI has requested a BIST 0 No BIST request pending May be read from the ADSP-21160 cluster bus only

Table 3-129
Chain Pointer Fetch End Description

Name	Type	Reset By	Reset Value	Function
Chain Pointer Fetch End addr 0x24[21]	Read Only	TR/ HTR	0	**Chain Descriptor Fetch Status** 1 The chain has started but the last descriptor has not been fetched yet. 0 Otherwise

Table 3-130
DMA Start/Done# Description

Name	Type	Reset By	Reset Value	Function
DMA Start/ Done#[4] addr 0x24[20]	Read/Write	TR/HTR	0	**Chain Mode Start/Done** Starts the Chain Mode DMA Controller. May be written or read from either the PCI bus or ADSP-21160 cluster bus. A simultaneous write results in the logical OR of the data bits. (If PCI bus writes a "1" and the ADSP-21160 cluster bus writes a "0" on the same clock cycle, a "1" get written, or vice-versa.) **Write:** 1 Start Chain Mode 0 No effect **Read:** 1 Chain Mode running 0 Chain Mode idle
(Sheet 1 of 3)				

Table 3-130
DMA Start/Done# Description (Continued)

Name	Type	Reset By	Reset Value	Function
DMA Start/ Done#[3] addr 0x24[19]	Read/Write	TR/HTR	0	**Receive Channel 1 Start/Done** Starts the Receive DMA Channel 1 controller. May be written or read from either the PCI bus or ADSP-21160 cluster bus. A simultaneous write results in the logical OR of the data bits. (If PCI bus writes a "1" and the ADSP-21160 cluster bus writes a "0" on the same clock cycle, a "1" get written, or vice-versa.) **Write:** 1 Start Receive DMA Channel 1 0 No Effect **Read:** 1 Receive DMA Channel 1 running 0 Receive DMA Channel 1 idle
DMA Start/ Done#[2] addr 0x24[18]	Read/Write	TR/HTR	0	**Receive Channel 0 Start/Done** Starts the Receive DMA Channel 0 controller. May be written or read from either the PCI bus or ADSP-21160 cluster bus. A simultaneous write results in the logical OR of the data bits. (If PCI bus writes a "1" and the ADSP-21160 cluster bus writes a "0" on the same clock cycle, a "1" get written, or vice-versa.) **Write:** 1 Start Receive DMA Channel 0 0 No Effect **Read:** 1 Receive DMA Channel 0 Running 0 Receive DMA Channel 0 Idle
(Sheet 2 of 3)				

Table 3-130
DMA Start/Done# Description (Continued)

Name	Type	Reset By	Reset Value	Function
DMA Start/ Done#[1] addr 0x24[17]	Read/Write	TR/HTR	0	**Transmit Channel 1 Start/Done** Starts the Transmit DMA Channel 1 controller. May be written or read from either the PCI bus or ADSP-21160 cluster bus. A simultaneous write results in the logical OR of the data bits. (If PCI bus writes a "1" and the ADSP-21160 cluster bus writes a "0" on the same clock cycle, a "1" get written, or vice-versa.) **Write:** 1 Start Transmit DMA Channel 1 0 No Effect **Read:** 1 Transmit DMA Channel 1 running 0 Transmit DMA Channel 1 idle
DMA Start/ Done#[0] addr 0x24[16]	Read/Write	TR/HTR	0	**Transmit Channel 0 Start/Done** Starts the Transmit DMA Channel 0 controller. May be written or read from either the PCI bus or ADSP-21160 cluster bus. A simultaneous write results in the logical OR of the data bits. (If PCI bus writes a "1" and the ADSP-21160 cluster bus writes a "0" on the same clock cycle, a "1" is written, or vice-versa.) **Write:** 1 Start Transmit DMA Channel 0 0 No Effect **Read:** 1 Transmit DMA Channel 0 running 0 Transmit DMA Channel 0 idle

(Sheet 3 of 3)

Table 3-131
PCI Incoming MB Status/User Outgoing MB Status

Name	Type	Reset By	Reset Value	Function
PCI Incoming MB Status[7:0] *User Outgoing MB Status[7:0]* addr 0x24[15:8]	Read/Write	TR/HTR	0	**PCI Incoming/User Outgoing Mail Status** 1 Mail data byte is full 0 Mail data byte is empty

Table 3-132
PCI Outgoing MB Status/User Incoming MB Status

Name	Type	Reset By	Reset Value	Function
PCI Outgoing MB Status[7:0] *User Incoming MB Status[7:0]* addr 0x24[7:0]	Read/Write	TR/HTR	0	**PCI Outgoing/User Incoming Mail Status** 1 Mail data byte is full 0 Mail data byte is empty

Table 3-133
Control Register Offset 0x26

Register Name: PCI_ID Register Offset: 0x26	
Bits	**Function**
63-56	*Single PCI Access Address Register[63:56]*
55-48	*Single PCI Access Address Register[55:48]*
47-40	*Single PCI Access Address Register[47:40]*
39-32	*Single PCI Access Address Register[39:32]*
31-24	*Single PCI Access Address Register[31:24]*
23-16	*Single PCI Access Address Register[23:16]*
15-8	*Single PCI Access Address Register[15:8]*
7-0	*Single PCI Access Address Register[7:0]*

Table 3-134
Single PCI Access Address Register Description

Name	Type	Reset By	Reset Value	Function
SPCI Access Address Register[63:0] addr 0x26[63:0]	Read/Write	TR/HTR	0	**Single PCI (SPCI) Access Target Address Register** Contains the target address for a Single PCI access. If a 32-bit access, upper 32 bits should be 0. May be accessed from the ADSP-21160 cluster bus only.

Table 3-135
Control Register Offset 0x28

Register Name: PCI_ID Register Offset: 0x28	
Bits	**Function**
63-56	*Single PCI Access Data Register[63:56]*
55-48	*Single PCI Access Data Register[55:48]*
47-40	*Single PCI Access Data Register[47:40]*
39-32	*Single PCI Access Data Register[39:32]*
31-24	*Single PCI Access Data Register[31:24]*
23-16	*Single PCI Access Data Register[23:16]*
15-8	*Single PCI Access Data Register[15:8]*
7-0	*Single PCI Access Data Register[7:0]*

Table 3-136
Single PCI Access Data Register Description

Name	Type	Reset By	Reset Value	Function
Single PCI Access Data Register[63:0] addr 0x28[63:0]	Read/Write	TR/HTR	0	**Single PCI (SPCI) Access Data Register.** On SPCI Write—data to be used for SPCI write operation. On SPCI Read—when SPCI completes, will contain the data resulting from the SPCI operation. May be accessed from the ADSP-21160 cluster bus only.

Table 3-137
Control Register Offset 0x2A

Register Name: PCI_ID Register Offset: 0x2A	
Bits	**Function**
63-56	*Reserved [63:56]*
55-48	*Reserved [55:48]*
47-40	*Reserved [47:40]*
39-32	*Reserved [39:32]*
31-24	*Reserved [31:24]*
23-16	*Reserved [23:16]*
15-8	*Reserved [15:8]*
7-0	*Reserved [7:0]*

Table 3-138
Reserved Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x2A[63:0]	N/A	N/A	0	**Reserved** Should be set to 0.

Table 3-139
Control Register Offset 0x2C

Register Name: PCI_ID Register Offset: 0x2C	
Bits	**Function**
63-56	*Receive FIFO0[63:56]*
55-48	*Receive FIFO0[55:48]*
47-40	*Receive FIFO0[47:40]*
39-32	*Receive FIFO0[39:32]*
31-24	*Receive FIFO0[31:24]*
23-16	*Receive FIFO0[23:16]*
15-8	*Receive FIFO0[15:8]*
7-0	*Receive FIFO0[7:0]*

Table 3-140
Receive FIFO0 Description

Name	Type	Reset By	Reset Value	Function
Receive FIFO0[63:0] addr 0x2C[63:0]	Read Only	TR/HTR	0	**Receive FIFO0** Contains the data presently at the output of Receive FIFO0. The FIFO will advance if the FIFO is read from and the *Dest_Rd* bit is set (0x40 bit[3]). May be accessed from the ADSP-21160 cluster bus only.

Table 3-141
Control Register Offset 0x2E

Register Name: PCI_ID Register Offset: 0x2E	
Bits	**Function**
63-56	*Receive FIFO1[63:56]*
55-48	*Receive FIFO1[55:48]*
47-40	*Receive FIFO1[47:40]*
39-32	*Receive FIFO1[39:32]*
31-24	*Receive FIFO1[31:24]*
23-16	*Receive FIFO1[23:16]*
15-8	*Receive FIFO1[15:8]*
7-0	*Receive FIFO1[7:0]*

Table 3-142
Receive FIFO1 Description

Name	Type	Reset By	Reset Value	Function
Receive FIFO1[63:0] addr 0x2E[63:0]	Read Only	TR/HTR	0	**Receive FIFO1** Contains the data presently at the output of Receive FIFO1. The FIFO will advance if the FIFO is read from and the Dest_Rd bit is set (0x40 bit[3]). May be accessed from the ADSP-21160 cluster bus only.

Table 3-143
Control Register Offset 0x30

Register Name: PCI_ID Register Offset: 0x30	
Bits	**Function**
63-56	*Transmit FIFO0[63:56]*
55-48	*Transmit FIFO0[55:48]*
47-40	*Transmit FIFO0[47:40]*
39-32	*Transmit FIFO0[39:32]*
31-24	*Transmit FIFO0[31:24]*
23-16	*Transmit FIFO0[23:16]*
15-8	*Transmit FIFO0[15:8]*
7-0	*Transmit FIFO0[7:0]*

Table 3-144
Transmit FIFO0 Description

Name	Type	Reset By	Reset Value	Function
Transmit FIFO0[63:0] addr 0x30[63:0]	Write Only	N/A	N/A	**Transmit FIFO0.** Data for Transmit FIFO0. The FIFO will accept the data and advance if the FIFO is written to. May be accessed from the ADSP-21160 cluster bus only.

Table 3-145
Control Register Offset 0x32

Register Name: PCI_ID Register Offset: 0x32	
Bits	**Function**
63-56	*Transmit FIFO1[63:56]*
55-48	*Transmit FIFO1[55:48]*
47-40	*Transmit FIFO1[47:40]*
39-32	*Transmit FIFO1[39:32]*

Table 3-145
Control Register Offset 0x32

Register Name: PCI_ID Register Offset: 0x32	
Bits	**Function**
31-24	*Transmit FIFO1[31:24]*
23-16	*Transmit FIFO1[23:16]*
15-8	*Transmit FIFO1[15:8]*
7-0	*Transmit FIFO1[7:0]*

Table 3-146
Transmit FIFO1 Description

Name	Type	Reset By	Reset Value	Function
Transmit FIFO1[63:0] addr 0x32[63:0]	Write Only	N/A	N/A	**Transmit FIFO1** Data for Transmit FIFO1. The FIFO will accept the data and advance if the FIFO is written to. May be accessed from the ADSP-21160 cluster bus only.

Table 3-147
Control Register Offset 0x34

Register Name: PCI_ID Register Offset: 0x34						
Bits	**Function**					
63-56	*0000_0000*					
55-48	*0000_00*					*DMA Arbitration Mode[1:0]*
47-40	*DMA Arbitration Priority[7:0]*					
39-32	*0000*			*DMA 32/64#[3:0]*		
31-24	*0000*			*DMA SPC[1:0]*		*00*
23-16	*0*	*Transmit FIFO Flush[1:0]*	*DMA Cancel[4:0]*			
15-8	*000*		*BIST Done [User]*	*BIST Code[3:0] [User Only]*		
7-0	*0*	*BE En[1:0]*	*Max Retry[1:0]*		*FIFO Thresh TO[1:0]*	*lat en*

Table 3-148
DMA Arbitration Mode Description

Name	Type	Reset By	Reset Value	Function
DMA ArbitrationMode[1:0] addr 0x34[49:48]	Read/Write	TR/HTR	0	**Master DMA Arbitration Mode** 00 Round Robin 01 Prioritized 10 Reserved 11 Reserved

Table 3-149
DMA Arbitration Priority

Name	Type	Reset By	Reset Value	Function
DMA Arbitration Priority[7:6] addr 0x34[47:46]	Read/Write	TR/HTR	0	**DMA Receive Channel 1 Arbitration Priority** 00 Low Priority 01 Medium Low Priority 10 Medium High Priority 11 High Priority
DMA Arbitration Priority[5:4] addr 0x34[45:44]	Read/Write	TR/HTR	0	**DMA Receive Channel 0 Arbitration Priority** 00 Low Priority 01 Medium Low Priority 10 Medium High Priority 11 High Priority
DMA Arbitration Priority[3:2] addr 0x34[43:42]	Read/Write	TR/HTR	0	**DMA Transmit Channel 1 Arbitration Priority** 00 Low Priority 01 Medium Low Priority 10 Medium High Priority 11 High Priority
DMA Arbitration Priority[1:0] addr 0x34[41:40]	Read/Write	TR/HTR	0	**DMA Transmit Channel 0 Arbitration Priority** 00 Low Priority 01 Medium Low Priority 10 Medium High Priority 11 High Priority

Table 3-150
DMA 32/64# Description

Name	Type	Reset By	Reset Value	Function
DMA 32/64#[3] addr 0x34[35]	Read/Write	TR/HTR	0	**Default DMA Channel Bus Width Receive Channel 1** 0 32 or 64 (auto detect) 1 32 bits only
DMA 32/64#[2] addr 0x34[34]	Read/Write	TR/HTR	0	**Default DMA Channel Bus Width Receive Channel 0** 0 32 or 64 (auto detect) 1 32 bits only
DMA 32/64#[1] addr 0x34[33]	Read/Write	TR/HTR	0	**Default DMA Channel Bus Width Transmit Channel 1** 0 32 or 64 (auto detect) 1 32 bits only
DMA 32/64#[0] addr 0x34[32]	Read/Write	TR/HTR	0	**Default DMA Channel Bus Width Transmit Channel 0** 0 32 or 64 (auto detect) 1 32 bits only

Table 3-151
Receive DMA Special Command Enable

Name	Type	Reset By	Reset Value	Function
DMA SPC[1] addr 0x34[27]	Read/Write	TR/HTR	0	**DMA Special Command Enable: Receive Channel 1** 0 Use only “memory read” (MR) 1 Use commands MR, MRM, MRL when appropriate.
DMA SPC[0] addr 0x34[26]	Read/Write	TR/HTR	0	**DMA Special Command Enable: Receive Channel 0** 0 Use only “memory read” (MR) 1 Use commands MR, MRM, MRL when appropriate.

Table 3-152
Transmit FIFO Flush Description

Name	Type	Reset By	Reset Value	Function
Transmit FIFO Flush[1] addr 0x34[22]	Read/Write	TR/HTR	0	**Transmit FIFO1 Flush** Flushes the Transmit FIFO1. The DMA operation must be idle. May be written or read from either the PCI bus or ADSP-21160 cluster bus. A simultaneous write results in the logical OR of the data bits. (If PCI bus writes a "1" and the ADSP-21160 cluster bus writes a "0" on the same clock cycle, a "1" gets written, and vice-versa.) **Write**: 1 Flush Transmit FIFO1 and related pipeline 0 No Effect **Read**: 1 Flushing operation is running 0 Flushing is complete
Transmit FIFO Flush[0] addr 0x34[21]	Read/Write	TR/HTR	0	**Transmit FIFO0 Flush** Flushes the Transmit FIFO0. The DMA operation must be idle. May be written or read from either the PCI bus or ADSP-21160 cluster bus. A simultaneous write results in the logical OR of the data bits. (If PCI bus writes a "1" and the ADSP-21160 cluster bus writes a "0" on the same clock cycle, a "1" gets written, and vice-versa.) **Write**: 1 Flush Transmit FIFO0 and related pipeline 0 No Effect **Read**: 1 Flushing operation is running 0 Flushing is complete

Table 3-153
DMA Cancel Description

Name	Type	Reset By	Reset Value	Function
DMA Cancel[4] addr 0x34[20]	Read/Write	TR/HTR	0	**DMA Cancel Chain Mode** 0 Cancel complete 1 Cancel chaining (write) Cancel not done (read)
DMA Cancel[3] addr 0x34[19]	Read/Write	TR/HTR	0	**DMA Cancel Master Receive Channel 1** 0 Cancel complete 1 Cancel DMA (write) Cancel not done (read)
DMA Cancel[2] addr 0x34[18]	Read/Write	TR/HTR	0	DMA Cancel Master Receive Channel 0 0 Cancel complete 1 Cancel DMA (write) Cancel not done (read)
DMA Cancel[1] addr 0x34[17]	Read/Write	TR/HTR	0	DMA Cancel Master Transmit Channel 1 0 Cancel complete 1 Cancel DMA (write) Cancel not done (read)
DMA Cancel[0] addr 0x34[16]	Read/Write	TR/HTR	0	**DMA Cancel Master Transmit Channel 0** 0 Cancel complete 1 Cancel DMA (write) Cancel not done (read)

Table 3-154
Built-In Self Test (BIST) Done Description

Name	Type	Reset By	Reset Value	Function
BIST Done [User] addr 0x34[12]	Write Only	TR/HTR	0	**Built-In Self Test Done** 0 BIST idle or running 1 BIST done

Table 3-155
Built-In Self Test (BIST) Completion Code Description

Name	Type	Reset By	Reset Value	Function
BIST Code[3:0] addr 0x34[11:8]	Read/Write	TR/HTR	0	**BIST Completion Code** 0 BIST OK 1–15 BIST failure

Table 3-156
Master Byte Enable Generation

Name	Type	Reset By	Reset Value	Function
Be En[1:0] addr 0x34[6:5]	Read/Write	TR/HTR	0	**Master Byte Enable Generation** 0 Automatic generation of Byte enables 1 Transmit FIFO Byte enables

Table 3-157
Maximum Master Retry Timeout Description

Name	Type	Reset By	Reset Value	Function
Max Retry[1:0] addr 0x34[4:3]	Read/Write	TR/HTR	0	**Maximum Master Retry Timeout** 00 = 30 µs 01 = 30 ms 10 = 30 s 11 = Infinite

Table 3-158
FIFO Threshold Timeout Description

Name	Type	Reset By	Reset Value	Function
FIFO Thresh TO[1:0] addr 0x34[2:1]	Read/Write	TR/HTR	0	**FIFO Threshold Timeout** 00 = 100 PCI clocks (**CLK**) 01 = 10,000 PCI clocks (**CLK**) 10 = 1,000,000 PCI clocks (**CLK**) 11 = Never Flush

Table 3-159
Latency Timeout Enable Description

Name	Type	Reset By	Reset Value	Function
lat en addr 0x34[0]	Read/Write	TR/HTR	1	**Master Latency Timeout Enable** 0 Latency Timeout Counter Enabled 1 Latency Timeout Counter Disabled

Table 3-160
Control Register Offset 0x36

Register Name: PCI_ID Register Offset: 0x36	
Bits	**Function**
63-56	*Reserved [63:56]*
55-48	*Reserved [55:48]*
47-40	*Reserved [47:40]*
39-32	*Reserved [39:32]*
31-24	*Reserved [31:24]*
23-16	*Reserved [23:16]*
15-8	*Reserved [15:8]*
7-0	*Reserved [7:0]*

Table 3-161
Reserved Register Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x36[63:0]	N/A	N/A	0	**Reserved** Must be set to 0.

Table 3-162
Control Register Offset 0x38

Register Name: PCI_ID Register Offset: 0x38	
Bits	**Function**
63-56	*Reserved [63:56]*
55-48	*Reserved [55:48]*
47-40	*Reserved [47:40]*
39-32	*Reserved [39:32]*
31-24	*Reserved [31:24]*
23-16	*Reserved [23:16]*
15-8	*Reserved [15:8]*
7-0	*Reserved [7:0]*

Table 3-163
Reserved Register Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x38[63:0	N/A	N/A	0	**Reserved** Must be set to 0.

Table 3-164
Control Register Offset 0x3A

Register Name: PCI_ID Register Offset: 0x3A		
Bits	**Function**	
63-56	*Reserved [30:23]*	
55-48	*Reserved [22:15]*	
47-40	*Reserved [14:7]*	
39-32	*Reserved [6:0]*	*Software Reset*
31-24	*Reserved [30:23]*	
23-16	*Reserved [22:15]*	
15-8	*Reserved [14:7]*	
7-0	*Reserved [6:0]*	*Total Reset*

Table 3-165
Reserved Register Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x3A[63:33] and addr 0x3A[31:1]	N/A	N/A	0	**Reserved** Must be set to 0.

Table 3-166
Software Reset Register Description

Name	Type	Reset By	Reset Value	Function
Software Reset addr 0x3A[32]	Write Only	TR/HTR	0	**Software Reset of ADSP-21160 Side** 1 Assert **fpga_reset** 0 Deassert **fpga_reset** May be accessed from the PCI only. 32-bit write only.

Table 3-167
Total Reset Register Description

Name	Type	Reset By	Reset Value	Function
Total Reset addr 0x3A[0]	Write Only	N/A	N/A	**Host Total Reset (HTR)** Executes a Host Total Reset (HTR). Self clearing. 1 Perform a Host Total Reset (HTR) 0 No effect May be accessed from the PCI only. 32-bit write only.

Table 3-168
Control Register Offset 0x3C

Register Name: PCI_ID Register Offset: 0x3C		
Bits	**Function**	
63-56	*Target Control Address[63:56]*	
55-48	*Target Control Address[55:48]*	
47-40	*Target Control Address[47:40]*	
39-32	*Target Control Address[39:32]*	
31-24	*Target Control Address[31:24]*	
23-16	*Target Control Address[23:16]*	
15-8	*Target Control Address[15:8]*	
7-0	*Target Control Address[7:3]*	*000*

Table 3-169
Target Control Address Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x3C[63:3]	Read Only	TR/HTR	0	**Reserved**

Table 3-170
Control Register Offset 0x3E

Register Name: PCI_ID Register Offset: 0x3E	
Bits	**Function**
63-56	*Target Control Data[63:56]*
55-48	*Target Control Data[55:48]*
47-40	*Target Control Data[47:40]*
39-32	*Target Control Data[39:32]*
31-24	*Target Control Data[31:24]*
23-16	*Target Control Data[23:16]*
15-8	*Target Control Data[15:8]*
7-0	*Target Control Data[7:0]*

Table 3-171
Target Control Data Description

Name	Type	Reset By	Reset Value	Function
Reserved addr 0x3E[63:0]	Read/Write	TR/HTR	*0*	**Reserved**

3.3.2 SHARC Interface Control Registers

This section describes the memory locations and settings for the SharcFIN's SHARC interface registers. All addresses described in this section are offsets from the base address of MS2 and are accessible from the ADSP-21160 DSPs and from the PCI interface. Table 3-172 gives the memory mapping for the user-configurable registers in the SharcFIN.

Note *Most of the user-configurable registers are already set and do not require you to program them. You will only need to set them if you are writing your own host interface programs.*

Table 3-172
Memory Map for the SHARC Interface Control Registers

Address	Register	Type	Description
0x40	Address Override	R/W	Allows addressing of IOP registers when ADSP-21160 is using a host packing mode
0x41	Status	R/O	General set of status registers. Indicates ADSP-21160 cluster bus status, and last reset source
0x42	Peripheral Bus Configuration	R/W	Configures and shows status of wait cycles of the 8-bit peripheral bus
0x43	Watchdog Configuration	WORM*	Enables and disables the watchdog timer
0x44	PMC+ Configuration	R/W	Configures the PMC+ interface
0x45	SD Size Config	R/W	Resets and reinitializes the SDRAM controller
0x46	Onboard I^2C Control	R/W	Controls the I^2C interface
0x47	PMC I^2C Control	R/W	Controls the I^2C interface to the PMC+ interface
0x48	DMA Address	R/W	Sets the address of DMA to be performed (writes into the register the address that the DMA is supposed to be operating on)
0x49	DMA Configuration	R/W	Controls various features of the SharcFIN DMA engine: size of DMA, increment size of DMA, other various configuration bits
0x4A	SD Window	R/W	Selects which 16 MB of SDRAM the PCI interface will view
0x4B–4F	Unused		
0x50,52, 54,56	H1I0, H2I0, H3I0, H4I0	R/W	Configure ADSP-21160 DSPs' interrupts; show status of interrupts
0x51,53, 55,57	Reserved		
0x58	PCInt	R/W	Configures PCI interrupt
0x5A	PMCI0	R/W	Configures PMC interrupt
0x59, 0x5B–5D	Unused		
0x5E	Flag Status	R/O	Shows state of all flags
0x5F	IRQ Status	R/O	Shows state of all interrupts

* Write Once Read Many

Address Override Register

The ***Address Override*** register (offset 0x40) configures how the ADSP-21160 DSPs access the least significant 32 bits on the ADSP-21160 cluster bus. It allows access to the DSPs' IOP space before the ADSP-21160's SYSCON register[1] has been configured.

Note *Only use this register if you are writing your own host interface programs.*

Table 3-173
Address Override Register Description

Bit	Name	Type	Reset By	Reset Value	Function
0	A0 Override En*	Read/Write		0	Address override enable for booting across the PCI bus
1	Overridden A0*	Write Only		0	Overridden address
2	BusLockReq	Write Only		0	Bus Lock Request. Requests the SharcFIN to acquire the ADSP-21160 cluster bus and locks access to the bus so that only the SharcFIN can access it. 0 = Disabled 1 = Requests that the SharcFIN acquire the ADSP-21160 cluster bus and not give it up
3	Destructive FIFO Read Enable	Write Only		1	Determines whether a read to the DMA FIFOs will cause the FIFOs to advance 0 = A read to the DMA FIFOs does not cause the FIFOs to advance 1 = A read to the DMA FIFOs causes the FIFOs to advance
4	Host Clock Disable	Read/Write		0	Disables the clock to the ADSP-21160 DSPs. This is used to address powerup anomalies on current editions of ADSP-21160 processors. 1 = Disables clock 0 = Clock enabled

* Do not use the Address Override bits (B0 and B1) under normal setup conditions. If you are running the board in standalone mode and are booting across the PCI bus, you can change these bits. However, exercise extreme caution since data loss or corruption will occur if you set the bits improperly.

1. The SYSCON register is a register in the ADSP-21160 DSPs that contains the MSIZE bits and is used to select the packing mode for synchronous and asynchronous transfers performed by the host.

Status Register

The ***Status*** register is a 16-bit read only register (offset 0x41) that gives information about various features on the board.

Table 3-174
Contents of the Status Register

Bit	Name	Type	Reset By	Reset Value	Function
0	PMCHostCfg	Read Only		Set in hardware	Indicates whether SharcFIN is configured for a PMC host site or a PMC daughter card. 1 = On baseboard 0 = On PMC
1	StandAlone	Read Only		Set by jumper	Determines whether PCI side resets are accepted by the SharcFIN 1 = PCI resets ignored 0 = PCI resets accepted
2	Bus Locked	Read Only		0	Indicates whether the SharcFIN has locked and acquired the ADSP-21160 cluster bus 0 = Cluster bus is not locked 1 = Cluster bus is locked
3	Last Reset Source	Read Only		0	Indicates whether the PCI interface or the watchdog was the source of the last board reset 0 = PCI reset 1 = Watchdog/external reset
4	SpareInput Pin	Read Only		1	Spare external input signal

Peripheral Bus Configuration Register

The ***Peripheral Bus Configuration*** register (offset 0x42) allows you to configure the wait cycles of the peripheral bus.

Table 3-175
Contents of the Peripheral Bus Configuration Register

Bit	Name	Type	Reset By	Reset Value	Function
3:0	PCI to Pbus Wait	Read/Write		0101 B0: 1 B1: 0 B2: 1 B3: 0	Select the number of wait cycles the SharcFIN will wait before completing a transaction on the peripheral bus. The actual value of wait cycles is one greater than the value in the register (for example, if the register value = 0, the number of wait cycles = 1). 0101 =Default setting (6 wait cycles)
4	Pbus Ack Enable	Read/Write		0	Selects whether the SharcFIN will monitor the peripheral bus Ack line after the peripheral bus wait time has expired. 0 = SharcFIN will wait the selected number of wait cycles and consider the transaction complete* 1 = SharcFIN will wait the selected number of wait cycles and then monitor the Ack line
5	Pbus Reset	Read/Write		0	Resets the peripheral bus reset line, the Flash, and the UART. The reset stays active until cleared by another write to the register.† 0 = No reset 1 = Resets Flash, UART, and all devices on the peripheral bus

* Five wait cycles is the minimum amount of wait cycles required to talk to the Flash memory.
† You can also reset the Flash, the UART, and all devices on the peripheral bus via a board reset.

Watchdog Configuration Register

The ***Watchdog Configuration*** register (offset 0x43) is a WORM (Write Once Read Many) register that allows you to enable or disable the watchdog timer, set its time-out time, and select which processor will reset its timer. Once the watchdog is enabled, it cannot be disabled except by a board reset (hence the WORM designation), which can be from the PCI interface, the watchdog, or an external source.

Table 3-176
Contents of the Watchdog Configuration Register

Bit	Name	Type	Reset By	Reset Value	Function
1:0	WDEn1, WDEn0	WORM		00	Enable the watchdog timer and select its time-out time. 00 = Disabled 10 = Enabled; short time-out (200 ms) 01 = Enabled; medium time-out (600 ms) 11 = Enabled; long time-out (1.2 s)
3:2	Unused				
7:4	H4F1 En, H3F1 En, H2F1 En, H1F1 En	WORM		0000	Selects which processor will strobe the watchdog timer.* 0001 = 21160-1 FLAG1 will strobe the watchdog 0010 = 21160-2 FLAG1 will strobe the watchdog 0100 = 21160-3 FLAG1 will strobe the watchdog 1000 = 21160-4 FLAG1 will strobe the watchdog

* You can select more than one processor, but it is not recommended.

PMC+ Configuration Register

The ***PMC+ Configuration*** register (offset 0x44) is a read/write register that configures the bus mode lines of the PMC+ interface and allows you to read their status. Table 3-177 below shows the contents of the ***PMC+ Configuration*** register.

Table 3-177
Contents of the PMC+ Configuration Register

Bit	Name	Type	Reset By	Reset Value	Function
0	PMC Flg/Int En	Read/Write		0	Configures the PMC+ interface's bus mode lines to be used as flag interrupts.* 1 = The PMC+ interface's bus mode lines will be used as flag interrupts 0 = The PMC+ interface's bus mode lines will be used as bus mode lines

Table 3-177
Contents of the PMC+ Configuration Register (Continued)

Bit	Name	Type	Reset By	Reset Value	Function
3:1	BusMode2, BusMode3, BusMode4	Read/Write		BusMode2: 1 BusMode3: 0 BusMode4: 0	Tells the PMC site whether or not it should drive BusMode1 to indicate its presence. When the flag/interrupts are disabled (by *PMC Flg/Int En* bit), the BusMode lines work according to the PMC specification. Bits 1–3 are Bus Mode lines 2–4. B1 = 1 Bus Mode line 2 B2 = 0 Bus Mode line 3 B3 = 0 Bus Mode line 4
4	BusMode 1	Read Only			Bus Mode line 1 is an input†. It indicates that a PMC card is present on the board.‡ 0 = PMC board is present
5	BusMode2	Read Only			Current status of BusMode2 line
6	BusMode3	Read Only			Current status of BusMode3 line
7	BusMode4	Read Only			Current status of BusMode4 line

* The option of using the bus mode lines as flag interrupts is a feature of BittWare's PMC+ form factor; to work properly, it must be enabled on both the PMC+ card and the host board.
† Bits 3:1 are outputs. Bit 4 is an input.
‡ Refer to the *IEEE P138.1 Standard Physical and Environmental Layers for PCI Mezzanine Cards: PMC* (PMC Specification) for details on the operation of these lines.

Onboard I²C Control Register

The ***Onboard I²C Control*** register (offset 0x46) controls the I²C interface. The I²C interface is a two-wire bus; one wire is a clock signal, and the other is a data signal. Both the clock and the data lines are pulled up. Table 3-178 shows the contents of the register.

As per standard I²C, both the clock and data lines are pulled high. Devices on the I²C bus either do not drive the bus, or they drive it low. Any device on the I²C bus can drive either the clock or data line low when required. You can also read the actual status of the lines.

Table 3-178
Contents of the Onboard I²C Control Register

Bit	Name	Type	Reset By	Reset Value	Function
0	Clock	Read/Write		1	On write, drives the clock line. On read, shows the state of the clock line
1	Data	Read/Write		1	On write, drives the data line. On read, shows the state of the data line
2	Clock Drive	Read Only		1	
3	Data Drive	Read Only		1	

When you write a 1 to either the clock or data line in this register, the SharcFIN does not drive the corresponding line. When you write a 0 to either the clock or the data line, the SharcFIN drives the corresponding line to 0. When you read either line, you read the actual state of the line rather than what you have written to it. If you are not driving the line, it will be 0 if another device is driving it and 1 if nothing is driving it. Table 3-179 shows the effect of the values written to the *Clock* and *Data* bits.

Table 3-179
Effects of Values Written to the Clock and Data Bits (B0, B1)

Value Written	Description
0	Drives the line low; when read back, shows 0
1	When read back, shows the actual state of the I²C line

PMC I²C Control Register

The ***PMC I_2C Control*** register (offset 0x47) controls the I²C interface to the PMC+ interface. The settings for this register are the same as the settings for the ***Onboard I²C*** register, except that all settings apply to the PMC+ interface I²C instead of the on-board I²C.

SDRAM Configuration Registers

The SharcFIN contains two registers that configure the SDRAM:

- ***SDRAM Size Configuration Register*** (offset 0x45)
- ***SDRAM Window Register*** (offset 0x4A)

SDRAM Size Configuration Register. The ***SDRAM Size Configuration*** register sets the size of the SDRAM. The settings for this register depend on the type of SDRAM modules used on the DSP board. Table 3-180 below shows the contents of the register.

Table 3-180
Contents of the SDRAM Size Configuration Register

Bit	Name	Type	Reset By	Reset Value	Function
1:0	SD Bank Size 1:0	Read/Write		0	Determine how the SDRAM controller uses the SharcFIN's CS0 and CS1 (chip select 0 and 1) pins. The chip select pins allow the SharcFIN to seamlessly connect to two banks of SDRAM. CS0 selects SDRAM bank 0, and CS1 selects SDRAM bank 1. **B0 B1 Result** 0 0 CS0 always active 0 1 CS1 active when 32-bit word address bit 24 is high; otherwise CS0 is active 1 0 CS1 active when 32-bit word address bit 25 is high; otherwise CS0 is active 1 1 CS1 active when 32-bit word address bit 26 is high; otherwise CS0 is active
2	SD RF Size	Read/Write		1	Sets the refresh rate of the SDRAM 0 = 4K refreshes every 64 milliseconds 1 = 8K refreshes every 64 milliseconds
3	SD Reset	Write Only		0	Writing a 1 resets the SDRAM controller and reinitializes the SDRAM

SDRAM Window Register. The ***SDRAM Window*** register is a 5-bit register that lets you select which 16 MB section of memory in the SDRAM to view from the host over the PCI interface. From the PCI side, the SDRAM is mapped into BAR4 with a 4 Mword (16 Mbyte) window viewable at a time. The offset into BAR4 provides bits 0 through 21 of the address of the SDRAM word to be accessed. The contents of the 5-bit ***SD Window*** register (address 0x4A) are appended to bits 22 through 26 of the address to complete the address and thereby select the 4 Mword window to be accessed. Table 3-181 lists the bits included in this register. For more information, refer to section 2.3.

Note *You will not need to configure this register since the Diag21k utility, which is included with the DSP21k-SF Toolkit, will set these bits.*

Table 3-181
Contents of the SDRAM Window Register

Bit	Name	Type	Reset By	Reset Value	Description
0	Window A0	Write Only		0	Selects window A0 of the SDRAM
1	Window A1	Write Only		1	Selects window A1 of the SDRAM
2	Window A2	Write Only		0	Selects window A2 of the SDRAM
3	Window A3	Write Only		0	Selects window A3 of the SDRAM
4	Window A4	Write Only		0	Selects window A4 of the SDRAM

DMA Address Register

The ***DMA Address*** register (offset 0x48) configures the address of the current DMA location. This register is incremented as the DMA progresses, allowing you to monitor the DMA engine's current address. You must reset this register each time if you wish to repeat a DMA on the same address range as last time. This register cannot be written to while the DMA start bit is set, but it can be read from at any time. The section entitled "Using the SharcFIN-Based DMA Engine" on page 62 describes this register in more detail.

Table 3-182
Contents of the DMA Address Register

Bit	Name	Type	Reset By	Reset Value	Function
0	Unused	Read Only			
27:1	A[27:1]	Read/Write		0	Indicates the current DMA location
31:28	Unused	Read Only			

DMA Configuration Register

The ***DMA Configuration*** register (offset 0x49) controls various features of the SharcFIN DMA engine, including starting a DMA, size of the DMA, increment size of the DMA, and other various configuration bits. Table 3-183 describes the contents of the register. The section entitled "Using the SharcFIN-Based DMA Engine" on page 62 explains this register in more detail. This register can not be written while the DMA start bit is set, but it can be read from at any time.

Table 3-183
Contents of the DMA Configuration Register

Bit	Name	Type	Reset By	Reset Value	Description
15:0	DMACntB[15:0]	Read/Write		X*	DMA transfer count (in 64-bit words) bits. All are settable.
22:16	DMA Stride B[6:0]	Read/Write		X	Stride or address increment bits. These bits will typically be set to 0x01.
23	Unused				Unwritable; fixed at 0.
24	DMAStart	Read/Write		0	Setting the bit to 1 starts the DMA; it resets to 0 when the DMA is complete.
25	DMA Channel Select	Read/Write		0	Selects the PCI channel being operated on (0 or 1)
26	DMA Direction	Read/Write		0	Selects whether a PCI transmit or receive DMA is being performed. 0 = PCI receive DMA (PCI to 21160/ SDRAM) 1 = PCI transmit DMA (21160/SDRAM to PCI)
27	DMA Interrupt	Read/Write		0	If this bit is set at the start of the DMA, the SharcFIN generates an interrupt in the interrupt multiplexer on completion of the DMA. Any write to this register clears the interrupt.
28	Burst Disable	Read/Write		0	1 = Disable bursting on the ADSP-21160 side. Must be set to 1 when the address increment > 0x01. If it is set when the address increment <= 0x01, it functions but will slow things down. 0 = Bursting enabled
29	DMA Buslock	Read/Write		0	1 = SharcFIN requests the ADSP-21160 cluster bus when the start bit is set, and once it obtains the bus, keeps it until the DMA completes.
(Sheet 1 of 2)					

Table 3-183
Contents of the DMA Configuration Register

Bit	Name	Type	Reset By	Reset Value	Description
31:30	DMA Xfer Length B[1:0]	Read/Write		00	Set the DMA transfer length. These bits must be set along with the bits in the PCI control register at 0x1A and 0x1B (0x68 byte address). **B30 B31 Result** 0 0 Data transferred during each access of the bus = eight 64-bit words 1 0 Data transferred during each access of the bus = sixteen 64-bit words 0 1 Data transferred during each access of the bus = thirty-two 64-bit words 1 1 Data transferred during each access of the bus = sixty-four 64-bit words Corresponding Receive FIFO almost empty or Transmit FIFO almost full flags must be set with corresponding value (length –1, so for B30, B31 must be set to 7).
(Sheet 2 of 2)					

* X = Unknown

Interrupt Configuration Registers

The SharcFIN features an interrupt multiplexer for each ADSP-21160, the PCI interface, and the PMC+ interface. Inputs to the multiplexers are flags from each ADSP-21160, a PCI side flag, PMC flags, UART flags, and a DMA flag. The registers at offsets 0x50 to 0x5A (see Table 3-184 on page 3-189) provide the interrupt multiplexers (see also section 2.4.2). Table 3-185 lists the settings for the ADSP-21160 interrupt multiplexers, Table 3-186 lists the settings for the PCI interrupt multiplexer, and Table 3-187 lists the settings for the PMC+ interrupt multiplexer.

The interrupt multiplexer registers are 32-bit registers that allow you to select the desired input sources. The first 16 bits [15:0] are read/write and select the source that will generate an interrupt to the processor; each of the bits corresponds to one of the flag inputs to the multiplexer. The second 16 bits [31:16] are read only and show which of the enabled interrupts are generating an interrupt, each bit corresponding to one of the flag inputs. Bits [31:16] are masked interrupt lines; when one of the flag inputs and its corresponding bit in bits [15:0] of the configuration register is high, the corresponding bit in bits [31:16] is also set to indicate the source of the input. Bits [31:16] are masked by 21160-1 IRQ0's interrupt mask.

Table 3-184
ADSP-21160 Interrupt Configuration Registers

Address	Register	Description
0x50	H1I0	Configures the operation of 21160-1 IRQ0
0x51	Unused	
0x52	H2I0	Configures the operation of 21160-2 IRQ0
0x53	Unused	
0x54	H3I0	Configures the operation of 21160-3 IRQ0
0x55	Unused	
0x56	H4I0	Configures the operation of 21160-4 IRQ0
0x57	Unused	
0x58	PCInt	Configures the operation of the PCI interrupt
0x59	Unused	
0x5A	PMCI0	Configures the operation of PMC+ IRQ0

Table 3-185
Settings for the ADSP-21160 Interrupt Configuration Registers (0x50, 0x52, 0x54, 0x56)

Bit	Name	Type	Reset Value	Description*
0	H1F0	Read/Write	0	Enables 21160-1 FLAG0 to cause an interrupt
1	H1F1	Read/Write	0	Enables 21160-1 FLAG1 to cause an interrupt
2	H2F0	Read/Write	0	Enables 21160-2 FLAG0 to cause an interrupt
3	H2F1	Read/Write	0	Enables 21160-2 FLAG1 to cause an interrupt
4	H3F0	Read/Write	0	Enables 21160-3 FLAG0 to cause an interrupt
5	H3F1	Read/Write	0	Enables 21160-3 FLAG1 to cause an interrupt
6	H4F0	Read/Write	0	Enables 21160-4 FLAG0 to cause an interrupt
7	H4F1	Read/Write	0	Enables 21160-4 FLAG1 to cause an interrupt
8	PCFlg	Read/Write	1	Enables the PCI interface to cause an interrupt
9	PMCFlg0	Read/Write	0	Enables FLAG0 from the PMC interface to cause an interrupt

(Sheet 1 of 2)

Table 3-185
Settings for the ADSP-21160 Interrupt Configuration Registers (0x50, 0x52, 0x54, 0x56)

Bit	Name	Type	Reset Value	Description*
10	Unused		0	
11	PRFlg	Read/Write	0	Enables a flag from the peripheral bus to cause an interrupt
12	UART0	Read/Write	0	Enables a flag from UART0 to cause an interrupt
13	UART1	Read/Write	0	Enables a flag from UART1 to cause an interrupt
14	DMAInterrupt	Read/Write	0	Enables a DMA interrupt
15	Unused		0	
16	H1F0	Read Only	0	Indicates that 21160-1 FLAG0 is generating an interrupt
17	H1F1	Read Only	0	Indicates that 21160-1 FLAG1 is generating an interrupt
18	H2F0	Read Only	0	Indicates that 21160-2 FLAG0 is generating an interrupt
19	H2F1	Read Only	0	Indicates that 21160-2 FLAG1 is generating an interrupt
20	H3F0	Read Only	0	Indicates that 21160-3 FLAG0 is generating an interrupt
21	H3F1	Read Only	0	Indicates that 21160-3 FLAG1 is generating an interrupt
22	H4F0	Read Only	0	Indicates that 21160-4 FLAG0 is generating an interrupt
23	H4F1	Read Only	0	Indicates that 21160-4 FLAG1 is generating an interrupt
24	PCFlg	Read Only	0	Indicates that a PCI flag is generating an interrupt
25	PMCFlg0	Read Only	0	Indicates that PMC+ FLAG0 is generating an interrupt
26	Unused		0	
27	PRFlg	Read Only	0	Indicates that a peripheral bus flag is generating an interrupt
28	UART0	Read Only	0	Indicates that UART0 flag is generating an interrupt
29	UART1	Read Only	0	Indicates that UART1 flag is generating an interrupt
30	DMAInterrupt	Read Only	0	Indicates that a DMA flag is generating an interrupt
31	Unused		0	

(Sheet 2 of 2)

* All descriptions in this column apply when bits are set to 1.

Table 3-186
Settings for the PCI Interrupt Configuration Register (0x58)

Bit	Name	Type	Reset Value	Description*
0	H1F0	Read/Write	1	Enables 21160-1 FLAG0 to cause an interrupt
1, 3, 5, 7	Unused		0	
2	H2F0	Read/Write	1	Enables 21160-2 FLAG0 to cause an interrupt
4	H3F0	Read/Write	1	Enables 21160-3 FLAG0 to cause an interrupt
6	H4F0	Read/Write	1	Enables 21160-4 FLAG0 to cause an interrupt
8	PCFlg	Read/Write	0	Enables the PCI interface to cause an interrupt
9	PMCFlg0	Read/Write	0	Enables FLAG0 from the PMC interface to cause an interrupt
10	Unused		0	
11	PRFlg	Read/Write	0	Enables a flag from the peripheral bus to cause an interrupt
12	UART0	Read/Write	0	Enables a flag from UART0 to cause an interrupt
13	UART1	Read/Write	0	Enables a flag from UART1 to cause an interrupt
14	DMAInterrupt	Read/Write	0	Enables a DMA interrupt
15, 17, 19, 21, 23	Unused		0	
16	H1F0	Read Only	0	Indicates that 21160-1 FLAG0 is generating an interrupt
18	H2F0	Read Only	0	Indicates that 21160-2 FLAG0 is generating an interrupt
20	H3F0	Read Only	0	Indicates that 21160-3 FLAG0 is generating an interrupt
22	H4F0	Read Only	0	Indicates that 21160-4 FLAG0 is generating an interrupt
24	PCFlg	Read Only	0	Indicates that a PCI flag is generating an interrupt
25	PMCFlg0	Read Only	0	Indicates that PMC+ FLAG0 is generating an interrupt
26	Unused		0	
27	PRFlg	Read Only	0	Indicates that a peripheral bus flag is generating an interrupt
28	UART0	Read Only	0	Indicates that UART0 flag is generating an interrupt
29	UART1	Read Only	0	Indicates that UART1 flag is generating an interrupt
30	DMAInterrupt	Read Only	0	Indicates that a DMA flag is generating an interrupt
31	Unused		0	

* All descriptions in this column apply when bits are set to 1.

Table 3-187
Settings for the PMC+ Interrupt Configuration Register (0x5A)

Bit	Name	Type	Reset Value	Description*
0	H1F0	Read/Write	0	Enables 21160-1 FLAG0 to cause an interrupt
1, 3, 5, 7	Unused		0	
2	H2F0	Read/Write	0	Enables 21160-2 FLAG0 to cause an interrupt
4	H3F0	Read/Write	0	Enables 21160-3 FLAG0 to cause an interrupt
6	H4F0	Read/Write	0	Enables 21160-4 FLAG0 to cause an interrupt
8	PCFlg	Read/Write	0	Enables the PCI interface to cause an interrupt
9	PMCFlg0	Read/Write	0	Enables FLAG0 from the PMC interface to cause an interrupt
10	Unused		0	
11	PRFlg	Read/Write	0	Enables a flag from the peripheral bus to cause an interrupt
12	UART0	Read/Write	0	Enables a flag from UART0 to cause an interrupt
13	UART1	Read/Write	0	Enables a flag from UART1 to cause an interrupt
14	DMAInterrupt	Read/Write	0	Enables a DMA interrupt
15, 17, 19, 21, 23	Unused		0	
16	H1F0	Read Only	0	Indicates that 21160-1 FLAG0 is generating an interrupt
18	H2F0	Read Only	0	Indicates that 21160-2 FLAG0 is generating an interrupt
20	H3F0	Read Only	0	Indicates that 21160-3 FLAG0 is generating an interrupt
22	H4F0	Read Only	0	Indicates that 21160-4 FLAG0 is generating an interrupt
24	PCFlg	Read Only	0	Indicates that a PCI flag is generating an interrupt
25	PMCFlg0	Read Only	0	Indicates that PMC+ FLAG0 is generating an interrupt
26	Unused		0	
27	PRFlg	Read Only	0	Indicates that a peripheral bus flag is generating an interrupt
28	UART0	Read Only	0	Indicates that UART0 flag is generating an interrupt
29	UART1	Read Only	0	Indicates that UART1 flag is generating an interrupt
30	DMAInterrupt	Read Only	0	Indicates that a DMA flag is generating an interrupt
31	Unused		0	

* All descriptions in this column apply when bits are set to 1.

Flag and Interrupt Status Registers

The registers at offsets 0x5E and 0x5F are 16-bit unmasked registers that show the status of all flags and interrupts. The register at 0x5E shows the status of the flags, and 0x5F shows the status of the interrupts. Table 3-188 and Table 3-189 describe the bits in the registers. For additional explanation of these registers, refer to the section entitled "Determining the Status of the Interrupts" on page 31.

Table 3-188
Contents of the Flag Status Register

Bit	Name	Type	Reset Value	Description
0	H1F0	Read Only	0	Status of 21160-1 FLAG0
1	H1F1	Read Only	0	Status of 21160-1 FLAG1
2	H2F0	Read Only	0	Status of 21160-2 FLAG0
3	H2F1	Read Only	0	Status of 21160-2 FLAG1
4	H3F0	Read Only	0	Status of 21160-3 FLAG0
5	H3F1	Read Only	0	Status of 21160-3 FLAG1
6	H4F0	Read Only	0	Status of 21160-4 FLAG0
7	H4F1	Read Only	0	Status of 21160-4 FLAG1
8	PCFlg	Read Only	0	Status of PCI flag
9	PMCFlg0	Read Only	0	Status of PMC+ FLAG0
10	Unused		0	
11	PRFlg	Read Only	0	Status of peripheral bus flag
12	UART0	Read Only	0	Status of UART0 flag
13	UART1	Read Only	0	Status of UART1 flag
14	DMAInterrupt	Read Only	0	Status of DMA flag
15	Unused		0	

Table 3-189
Contents of the Interrupt Status Register

Bit	Name	Type	Reset Value	Description
0	H1I0	Read Only	0	Status of 21160-1 IRQ0
1	H1I1	Read Only	0	Status of 21160-1 IRQ1
2	H2I0	Read Only	0	Status of 21160-2 IRQ0
3	H2I1	Read Only	0	Status of 21160-2 IRQ1
4	H3I0	Read Only	0	Status of 21160-3 IRQ0
5	H3I1	Read Only	0	Status of 21160-3 IRQ1
6	H4I0	Read Only	0	Status of 21160-4 IRQ0
7	H4I1	Read Only	0	Status of 21160-4 IRQ1
8	PCInt	Read Only	0	Status of PCI interrupt
9	PMCI0	Read Only	0	Status of PMC+ IRQ0
10	PMCI1	Read Only	0	Status of PMC+ IRQ1
15:11	Unused	Read Only	0	

Chapter 4
Hardware Description

This chapter describes the physical characteristics of the SharcFIN chip. It includes the following information:

- A detailed description of each signal (section 4.1)
- The chip's pinout (section 4.2)
- The chip's layout and dimensions (section 4.3)

4.1 SharcFIN Signal Descriptions

The following signal types are taken from the PCI Bus Specification:

- **in** Input is a standard input-only signal
- **out** Totem Pole Output is a standard active driver
- **t/s** Tri-State[1] is a bi-directional, tri-state input/output pin
- **s/t/s** Sustained tri-state is an active low tri-state signal owned and driven by one agent at a time. The agent that drives an s/t/s pin low must drive it high for at least one clock before letting it float. A new agent cannot start driving an s/t/s signal any sooner than one clock after the previous owner tri-states it. A pull-up is required to sustain the inactive state until another agent drives it, and must be provided by the central source.
- **o/d** Open Drain allows multiple devices to share as wire-OR

Note *For active low signals, the signal name is designated by an overline. For example, the $\overline{TRDY}$ signal is asserted low when the target is ready to complete a data transfer. Signals that are not designated by an overline are asserted when they assume the logic high state.*

Table 4-1
PCI Interface Signal Definitions

Signal	Type	Description
AD[63:0]	t/s	**Address/Data** Address and data are multiplexed on the same PCI bus pins. A PCI bus transaction consists of an address phase followed by one or more data phases. An address phase occurs on the first or first+second **CLK** cycle in which **$\overline{\text{FRAME}}$** is asserted. A data transfer occurs on the CLK cycles in which **$\overline{\text{IRDY}}$** and **TRDY** are both asserted.
(Sheet 1 of 5)		

1. Tri-State is a Registered Trademark of National Semiconductor.

Table 4-1
PCI Interface Signal Definitions (Continued)

Signal	Type	Description
C/BE[7:0]	t/s	**Command/Byte Enable** Bus commands and byte enables are multiplexed on the same pins. These pins define the current bus command during an address phase. During a data phase, these pins are used as Byte Enables, with **C/BE[0]** enabling byte 0 (Least Significant Byte (LSB)) and **C/BE[7]** enabling byte 7 (Most Significant Byte (MSB)). The commands are active high, and the byte enables are active low. **C/BE[3:0] — Command Type** 0b0000 — Interrupt Acknowledge 0b0001 — Special Cycle 0b0010 — I/O Read 0b0011 — I/O Write 0b0100 — Reserved 0b0101 — Reserved 0b0110 — Memory Read 0b0111 — Memory Write 0b1000 — Reserved 0b1001 — Reserved 0b1010 — Configuration Read 0b1011 — Configuration Write 0b1100 — Memory Read Multiple 0b1101 — Dual Address Cycle (DAC) 0b1110 — Memory Read Line 0b1111 — Memory Write and Invalidate
PAR	t/s	**PCI Parity Line** Parity is always driven as even from all **AD[31:0]** and **C/BE[3:0]** signals. The parity is valid during the clock following the address phase and is driven by the bus master. During a data phase for write transactions, the bus master sources this signal on the clock following **IRDY** active. During a data phase for read transactions, this signal is driven by the target and is valid on the clock following **TRDY** active. The **PAR** signal has the same timing as **AD[31:0]**, delayed by one clock.
PCICLK	in	**PCI Clock** The rising edge of this signal is the reference upon which all other PCI signals are based except for **RST** and **INTA.** The maximum CLK frequency for the PCI portion of the SharcFIN is 75 MHz, and the minimum is DC (0 Hz). The SharcFIN is a fully static device; and since no phase-lock-loops (PLLs) are used on CLK, the SharcFIN supports all forms of clock gating techniques, such as spread spectrum and clock run.
(Sheet 2 of 5)		

Table 4-1
PCI Interface Signal Definitions (Continued)

Signal	Type	Description
RST	in	**PCI Reset Signal** Reset brings the SharcFIN to a known state: • All PCI bus output signals tri-stated • All open drain signals (e.g. **SERR**) floated • All registers set to their factory defaults • Master and Target are returned to an idle state • All FIFOs emptied • User-reset is asserted • Automatic 32/64-bit PCI bus sizing on rising edge
FRAME	s/t/s	**PCI Frame** This signal is driven by the current bus master to indicate the beginning and duration of a bus transaction. When **FRAME** is first asserted, it indicates a bus transaction is beginning with valid addresses and bus command present on **AD[63:0]** and **C/BE[7:0]** (or **AD[31:0]** and **C/BE[3:0]**). Data transfers continuing while **FRAME** is deasserted indicate the transaction is in a final data phase or has completed.
IRDY	s/t/s	**PCI Initiator Ready** This signal is always driven by the bus master to indicate its ability to complete the current data phase. During write transactions, it indicates **AD[63:0]** (or **AD[31:0]**) contains valid data.
TRDY	s/t/s	**PCI Target Ready** This signal is driven by the selected target to indicate the target is able to complete the current data phase. During read transactions, it indicates **AD[63:0]** (or **AD[31:0]**) contains valid data. Wait states occur until both **TRDY** and **IRDY** are asserted together.
STOP	s/t/s	**PCI Stop** The **STOP** signal is driven by a selected target and conveys a request to the bus master to stop the current transaction.
IDSEL	in	**PCI Initialization Device Select** This pin is used as a chip select during configuration read or write transactions. It is valid only during first address phase.
DEVSEL	s/t/s	**PCI Device Select** This signal is driven by a target decoding and recognizing one of its bus addresses. It informs a bus master that an agent has decoded a current bus cycle.
INTA	o/d	**PCI Interrupt A** This signal is defined as optional and level-sensitive. Driving it low will interrupt the host. The **INTA** interrupt is to be used for any single function device requiring an interrupt capability.
(Sheet 3 of 5)		

Table 4-1
PCI Interface Signal Definitions (Continued)

Signal	Type	Description
PERR	s/t/s	**PCI Parity Error Detected** Only for reporting data parity errors for all bus transactions except for Special Cycles. It is driven by the agent receiving data two clock cycles after the data phase when parity was detected to be in error. This signal is driven inactive (high) for one clock cycle prior to returning to the tri-state condition.
SERR	o/d	**PCI System Error** Used to report address and data parity errors on Special Cycle commands and any other error condition having a catastrophic system impact. Special Cycle commands are not received by the SharcFIN.
REQ64	s/t/s	**PCI Request 64-bit Transfer** When asserted by the current bus master, indicates it desires to transfer data using 64 bits. **REQ64** has the same timing as **FRAME**. **REQ64** also has meaning at the end of the reset as described in section 2.6.
ACK64	s/t/s	**PCI Acknowledge 64-bit Transfer** When actively driven by the device that has positively decoded its address as the target of the current access, indicates the target is willing to transfer data using 64 bits. **ACK64** has the same timing as **DEVSEL.** **Only asserted if REQ64 is asserted in the same transaction.**
PAR64	t/s	**Parity Upper (PCI 64-bit parity line)** The Parity Upper dword is the even parity bit that protects **AD[63:32]** and **C/BE[7:4]**. **PAR64** is valid one clock after the initial address phase when **REQ64** is asserted and the DAC command is indicated on **C/BE[3:0]. PAR64** is also valid the clock after the second address phase of a DAC command when **REQ64** is asserted. **PAR64** is stable and valid for 64-bit data phases one clock after either **IRDY** is asserted on a write transaction or **TRDY** is asserted on a read transaction. **PAR64** has the same timing as **AD[63:32]**, but delayed by one clock. The master drives **PAR64** for address and write data phases. The target drives **PAR64** for read data phases.
REQ	out/t/s	**PCI Request** Request indicates to the arbiter that this agent desires use of the bus. This is a point-to-point signal. Every master has its own **REQ** which must be tri-stated while **RST** is asserted. **REQ** is always driven after reset.
GNT	in	**Grant** Grant indicates to the agent that access to the bus has been granted. This is a point-to-point signal. Every master has its own **GNT** which must be ignored while **RST** is asserted.
TDI	in	**Test Data Input** Used to serially shift test data and test instructions into the device during Test Access Port (TAP) operation.

(Sheet 4 of 5)

Table 4-1
PCI Interface Signal Definitions (Continued)

Signal	Type	Description
TDO	out	**Test Output** Used to serially shift test data and test instructions out of the device during TAP operation.
TCK	in	**Test Clock** Used to clock state information and test data into and out of the device during the operation of the TAP.
TMS	in	**Test Mode Select** Used to control the state of the TAP controller in the device.
$\overline{\text{TRST}}$	in	**Test Reset** Provides an asynchronous initialization of the TAP controller. This signal is optional in IEEE Standard 1149.1.
(Sheet 5 of 5)		

Table 4-2
SHARC Interface Signal Definitions

Signal	Type	Description
ADSP-21160 Signals		
$\overline{\text{H\_WrH}}$	t/s	**ADSP-21160 High Word Write** Connects directly to the ADSP-21160 cluster bus **$\overline{\text{WrH}}$** line.
$\overline{\text{H\_Wrl}}$	t/s	**ADSP-21160 Low Word Write** Connects directly to the ADSP-21160 cluster bus **$\overline{\text{Wrl}}$** line.
$\overline{\text{H\_RdH}}$	t/s	**ADSP-21160 High Word Read** Connects directly to the ADSP-21160 cluster bus **$\overline{\text{RdH}}$** line.
$\overline{\text{H\_RdL}}$	t/s	**ADSP-21160 Low Word Read** Connects directly to the ADSP-21160 cluster bus **$\overline{\text{RdL}}$** line.
H_Data[63:0]	t/s	**ADSP-21160 Data Bits** Connect directly to the ADSP-21160 cluster bus **Data[63:0]** lines.
H_Addr[31:0]	t/s	**ADSP-21160 Address Bits** Connect directly to the ADSP-21160 cluster bus **Addr[31:0]** lines.
H_Ack	t/s	**ADSP-21160 Ack Line** Connects directly to the ADSP-21160 cluster bus **Ack** line.
(Sheet 1 of 6)		

Table 4-2
SHARC Interface Signal Definitions (Continued)

Signal	Type	Description
H_Brst	t/s	**ADSP-21160 Burst Signal** Connects directly to the ADSP-21160 cluster bus **Brst** line.
$\overline{\text{H\_HBR}}$	out	**ADSP-21160 Host Bus Request** Connects directly to the ADSP-21160 cluster bus **$\overline{\text{HBR}}$** line.
$\overline{\text{H\_HBG}}$	in	**ADSP-21160 Host Bus Grant** Connects directly to the ADSP-21160 cluster bus **$\overline{\text{HBG}}$** line.
$\overline{\text{H\_SBTS}}$	out	**ADSP-21160 Suspend/Bus Tristate** Connects directly to the ADSP-21160 cluster bus **$\overline{\text{SBTS}}$** line.
$\overline{\text{H\_MS0}}$	t/s	**ADSP-21160 Memory Select 0 (SDRAM)** Gets passed to MS (memory select) lines on SDRAM. Connects directly to the ADSP-21160 cluster bus **$\overline{\text{MS0}}$** line.
$\overline{\text{H\_MS1}}$	t/s	**ADSP-21160 Memory Select 1 (Flash, UART, Peripheral Bus)** Gets passed through to **PR_FlashSel**, **PR_UART**, and **PR_UBSEL** by SharcFIN. Connects directly to the ADSP-21160 cluster bus **$\overline{\text{MS1}}$** line.
$\overline{\text{H\_MS2}}$	t/s	**ADSP-21160 Memory Select 2 (SharcFIN)** Selects SharcFIN for target operation by an ADSP-21160. Connects directly to the ADSP-21160 cluster bus **$\overline{\text{MS2}}$** line.
$\overline{\text{H\_MS3}}$	t/s	**ADSP-21160 Memory Select 3 (Unused)** Unused. Connects directly to the ADSP-21160 cluster bus **$\overline{\text{MS3}}$** line.
$\overline{\text{H\_BMS}}$	in	**ADSP-21160 Boot Memory Select** Input to SharcFIN. Gets passed through to **PR_FlashSel** by SharcFIN. Connects directly to the ADSP-21160 cluster bus **$\overline{\text{BMS}}$** line.
H_Page	in	**ADSP-21160 External Memory Page Miss** Connects directly to the ADSP-21160 **Page** line.
$\overline{\text{H1\_DMAR}}$	out	**ADSP-21160-1 DMA Request 1 Line** Connects directly to the ADSP-21160-1 **$\overline{\text{DMAR}}$** line.
$\overline{\text{H2\_DMAR}}$	out	**ADSP-21160-2 DMA Request 1 Line** Connects directly to the ADSP-21160-2 **$\overline{\text{DMAR}}$** line.
$\overline{\text{H3\_DMAR}}$	out	**ADSP-21160-3 DMA Request 1 Line** Connects directly to the ADSP-21160-3 **$\overline{\text{DMAR}}$** line.
$\overline{\text{H4\_DMAR}}$	out	**ADSP-21160-4 DMA Request 1 Line** Connects directly to the ADSP-21160-4 **$\overline{\text{DMAR}}$** line.
$\overline{\text{H1\_IRQ0}}$	out	**ADSP-21160-1 IRQ0** Connects directly to the ADSP-21160-1 **$\overline{\text{IRQ0}}$** line.
(Sheet 2 of 6)		

Table 4-2
SHARC Interface Signal Definitions (Continued)

Signal	Type	Description
$\overline{\text{H2\_IRQ0}}$	out	**ADSP-21160-2 IRQ0** Connects directly to the ADSP-21160-2 **$\overline{\text{IRQ0}}$** line.
$\overline{\text{H3\_IRQ0}}$	out	**ADSP-21160-3 IRQ0** Connects directly to the ADSP-21160-3 **$\overline{\text{IRQ0}}$** line.
$\overline{\text{H4\_IRQ0}}$	out	**ADSP-21160-4 IRQ0** Connects directly to the ADSP-21160-4 **$\overline{\text{IRQ0}}$** line.
$\overline{\text{H1\_IRQ1}}$	out	**ADSP-21160-1 IRQ1** Connects directly to the ADSP-21160-1 **$\overline{\text{IRQ1}}$** line. Its output is **H3_Flag1** AND **H4_Flag1**. (Either flag low generates an interrupt.)
$\overline{\text{H2\_IRQ1}}$	out	**ADSP-21160-2 IRQ1** Connects directly to the ADSP-21160-2 **$\overline{\text{IRQ1}}$** line. Its output is **H3_Flag1** AND **H4_Flag1**. (Either flag low generates an interrupt.)
$\overline{\text{H3\_IRQ1}}$	out	**ADSP-21160-3 IRQ1** Connects directly to the ADSP-21160-3 **$\overline{\text{IRQ1}}$** line. Its output is **H1_Flag1** AND **H2_Flag1**. (Either flag low generates an interrupt.)
$\overline{\text{H4\_IRQ1}}$	out	**ADSP-21160-4 IRQ1** Connects directly to the ADSP-21160-4 **$\overline{\text{IRQ1}}$** line. Its output is **H1_Flag1** AND **H2_Flag1.** (Either flag low generates an interrupt.)
H1_Flag0	in	**ADSP-21160-1 Flag0** Connects directly to the ADSP-21160-1 **FLAG0** line. Generates an interrupt in the interrupt multiplexer when low and enabled.
H2_Flag0	in	**ADSP-21160-2 Flag0** Connects directly to the ADSP-21160-2 **FLAG0** line. Generates an interrupt in the interrupt multiplexer when low and enabled.
H3_Flag0	in	**ADSP-21160-3 Flag0** Connects directly to the ADSP-21160-3 **FLAG0** line. Generates an interrupt in the interrupt multiplexer when low and enabled.
H4_Flag0	in	**ADSP-21160-4 Flag0** Connects directly to the ADSP-21160-4 **FLAG0** line. Generates an interrupt in the interrupt multiplexer when low and enabled.
H1_Flag1	in	**ADSP-21160-1 Flag1** Connects directly to the ADSP-21160-1 **FLAG1** line.
H2_Flag1	in	**ADSP-21160-2 Flag1** Connects directly to the ADSP-21160-2 **FLAG1** line.
(Sheet 3 of 6)		

Table 4-2
SHARC Interface Signal Definitions (Continued)

Signal	Type	Description
H3_Flag1	in	**ADSP-21160-3 Flag1** Connects directly to the ADSP-21160-3 **FLAG1** line.
H4_Flag1	in	**ADSP-21160-4 Flag1** Connects directly to the ADSP-21160-4 **FLAG1** line.
$\overline{\text{ResetO}}$	out	**ADSP-21160 Reset** Connects to **$\overline{\text{Reset}}$** line on all ADSP-21160 processors.
Clock Signals		
ClkEn	out	**ADSP-21160 Clock Enable** Enables clocks to all ADSP-21160s to fix their PLL power-up issue. Active low (low should enable clocks).
LClk	in	**Clock to FPGA** Must be same frequency as ADSP-21160s and must not be disabled by **ClkEn**.
Spare Input	in	**Unused FPGA Pin** Pull up if unused. Can be read at configuration register 0x41 bit[4].
Peripheral Bus Signals		
PR_Addr[20:0]	out	**Peripheral Bus Address Bits**
PR_Data[7:0]	t/s	**Peripheral Bus Data Bits**
PR_ACK	in	**Peripheral Bus Ack Line**
$\overline{\text{PR\_FLASHSel}}$	out	**Peripheral Bus Flash Select** Connects to boot Flash select line.
$\overline{\text{PR\_UARTSel}}$	out	**Peripheral Bus UART Select** Connects to UART select line.
$\overline{\text{PR\_Wr}}$	out	**Peripheral Bus Write**
$\overline{\text{PR\_Int}}$	in	**Peripheral Bus Interrupt Line**
$\overline{\text{PR\_Rst}}$	out	**Peripheral Bus Reset**
$\overline{\text{PR\_Rd}}$	out	**Peripheral Bus Read**
$\overline{\text{PR\_UBSel}}$	out	**Peripheral Bus Connector Select Line**
UARTInt0	in	**Peripheral Bus UART Interrupt Line 0**
(Sheet 4 of 6)		

Table 4-2
SHARC Interface Signal Definitions (Continued)

Signal	Type	Description
UARTInt1	in	**Peripheral Bus UART Interrupt Line 1**
UART_Rst	out	**Peripheral Bus UART Reset**
SDRAM Signals		
SD_A10	out	**SDRAM A10 Line**
SD_AddrMux	out	**SDRAM Address Mux Line** Switches in row and column address lines.
SD_CKE	out	**SDRAM Clock Enable**
SD_DQML	out	**SDRAM Low Word Data Mask Line**
SD_DQMH	out	**SDRAM High Word Data Mask Line**
$\overline{\text{SD\_WE}}$	out	**SDRAM Write Enable**
$\overline{\text{SD\_RAS}}$	out	**SDRAM RAS**
$\overline{\text{SD\_CAS}}$	out	**SDRAM CAS**
$\overline{\text{SD\_CS0}}$	out	**SDRAM Bank 0 Chip Select**
$\overline{\text{SD\_CS1}}$	out	**SDRAM Bank 1 Chip Select**
Watchdog Signals		
$\overline{\text{WD\_Tcl}}$	out	**Watchdog Tickle Signal**
$\overline{\text{WD\_Rst}}$	in	**External Reset/Watchdog Reset Input**
WD_TDly	t/s	**Watchdog Tickle Delay**
Standalone Signals		
$\overline{\text{StAlone}}$	in	**Standalone Signal** Disables PCI interface when pulled low.
PMC Interface Signals		
$\overline{\text{PA\_BusM1}}$	t/s	PMC Site Bus Mode 1
$\overline{\text{PA\_BusM2}}$	t/s	PMC Site Bus Mode 2
(Sheet 5 of 6)		

Table 4-2
SHARC Interface Signal Definitions (Continued)

Signal	Type	Description
$\overline{\text{PA\_BusM3}}$	t/s	PMC Site Bus Mode 3
$\overline{\text{PA\_BusM4}}$	t/s	PMC Site Bus Mode 4
PMC_Host	in	**PMC Host Select Line** High = PMC host; Low = PMC card
I^2C Interface Signals		
P_SCL	t/s	**Secondary I^2C Bus Clock Line** Pull up
P_SDA	t/s	**Secondary I^2C Bus Data Line** Pull up
H_SDA	t/s	**SharcFIN Configuration EEPROM Clock Line** Pull up
H_SCL	out	**SharcFIN Configuration EEPROM Data Line** Pull up
H_EEPWP	out	**SharcFIN Configuration EEPROM Write Protect**
(Sheet 6 of 6)		

4.2 Pinout

Table 4-3
SharcFIN Pinout

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
A1	NC	B5	REQ	C9	H_Ack	D13	NC	E17	P33V
A2	NC	B6	AD[18]	C10	H_Brst	D14	H_Addr[01]	E18	P33V
A3	AD[29]	B7	AD[22]	C11	H_MS0	D15	H2_DMAR	E19	SD_CAS
A4	AD[27]	B8	RST	C12	H_Data[00]	D16	PR_Rd	E20	H_Data[40]
A5	AD[30]	B9	V(I/O)	C13	H_Data[15]	D17	NC	E21	H_Data[19]
A6	IDSEL	B10	H_RdH	C14	H_SBTS	D18	NC	E22	H_Data[26]
A7	AD[26]	B11	H_Data[10]	C15	PR_Wr	D19	GND	F1	DEVSEL
A8	NC	B12	H_Data[09]	C16	ClkEn	D20	PR_Data[6]	F2	NC
A9	H_Wrl	B13	H_Data[14]	C17	H_HBG	D21	H_Data[24]	F3	AD[17]
A10	H_Data[07]	B14	PR_Ack	C18	H_Addr[07]	D22	H_Data[41]	F4	NC
A11	H_Data[04]	B15	H_HBR	C19	WD_Tcl	E1	PERR	F5	NC
A12	H_Data[12]	B16	PR_Addr[01]	C20	PR_Data[7]	E2	AD[15]	F6	GND
A13	H_Data[11]	B17	H_Addr[06]	C21	SD_AddrMux	E3	AD[25]	F7	P33V
A14	H_Data[13]	B18	PR_Addr[07]	C22	H_Data[42]	E4	AD[23]	F8	GND
A15	SD_DQMH	B19	PR_Addr[05]	D1	C/BE[2]	E5	NC	F9	GND
A16	PR_Addr[08]	B20	TCK	D2	PAR	E6	P33V	F10	GND
A17	PR_Addr[06]	B21	NC	D3	AD[19]	E7	AD[20]	F11	H_Data[05]
A18	SD_RAS	B22	NC	D4	C/BE[3]	E8	AD[28]	F12	H_Data[02]
A19	H_Addr[05]	C1	AD[21]	D5	NC	E9	NC	F13	GND
A20	PR_Data[5]	C2	NC	D6	NC	E10	H_RdL	F14	ResetO
A21	NC	C3	NC	D7	AD[24]	E11	P33V	F15	GND
A22	NC	C4	NC	D8	NC	E12	H_Data[01]	F16	P33V
B1	NC	C5	AD[31]	D9	NC	E13	H_Data[17]	F17	H_Data[18]
B2	NC	C6	GNT	D10	NC	E14	SD_DQML	F18	SD_WE
B3	TDO	C7	NC	D11	H_Data[08]	E15	H1_DMAR	F19	H_Data[36]
B4	AD[16]	C8	V(I/O)	D12	H_Data[03]	E16	H_Addr[08]	F20	H_Data[23]

(Sheet 1 of 4)

Table 4-3
SharcFIN Pinout (Continued)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
F21	H_Data[31]	H6	AD[14]	J13	GND	K20	H_Data[51]	M5	PCICLK
F22	H_Data[30]	H7	P33V	J14	GND	K21	H_Data[56]	M6	GND
G1	$\overline{\text{C/BE[1]}}$	H8	GND	J15	P33V	K22	H_Data[55]	M7	GND
G2	AD[13]	H9	P33V	J16	GND	L1	AD[06]	M8	P33V
G3	NC	H10	P33V	J17	H_Data[49]	L2	StAlone	M9	GND
G4	$\overline{\text{SERR}}$	H11	H_Data[06]	J18	H_Data[45]	L3	PMC_Host	M10	GND
G5	GND	H12	GND	J19	NC	L4	NC	M11	GND
G6	P33V	H13	NC	J20	H_Data[54]	L5	$\overline{\text{WD\_Rst}}$	M12	GND
G7	GND	H14	P33V	J21	H_Data[46]	L6	$\overline{\text{ACK64}}$	M13	GND
G8	$\overline{\text{INTA}}$	H15	GND	J22	H_Data[53]	L7	GND	M14	GND
G9	GND	H16	H_Data[29]	K1	$\overline{\text{TRDY}}$	L8	P33V	M15	P33V
G10	$\overline{\text{H\_WrH}}$	H17	H_Data[22]	K2	NC	L9	GND	M16	GND
G11	GND	H18	H_Data[27]	K3	$\overline{\text{FRAME}}$	L10	GND	M17	UARTInt0
G12	NC	H19	NC	K4	$\overline{\text{STOP}}$	L11	GND	M18	UARTInt1
G13	H_Data[16]	H20	H_Data[35]	K5	$\overline{\text{IRDY}}$	L12	GND	M19	NC
G14	GND	H21	H_Data[44]	K6	P33V	L13	GND	M20	LClk
G15	GND	H22	H_Data[48]	K7	P33V	L14	GND	M21	Spare Input
G16	GND	J1	AD[09]	K8	GND	L15	P33V	M22	H_Data[60]
G17	H_Data[21]	J2	C/BE[0]	K9	GND	L16	GND	N1	AD[08]
G18	H_Data[20]	J3	NC	K10	GND	L17	H_Data[62]	N2	NC
G19	H_Data[28]	J4	AD[10]	K11	GND	L18	H_Data[58]	N3	AD[07]
G20	H_Data[25]	J5	GND	K12	GND	L19	H_Data[59]	N4	AD[05]
G21	H_Data[38]	J6	GND	K13	GND	L20	H_Data[52]	N5	P33V
G22	H_Data[43]	J7	GND	K14	GND	L21	H_Data[57]	N6	$\overline{\text{REQ64}}$
H1	AD[12]	J8	P33V	K15	GND	L22	H_Data[63]	N7	P33V
H2	NC	J9	GND	K16	H_Data[47]	M1	AD[04]	N8	GND
H3	AD[11]	J10	GND	K17	H_Data[37]	M2	NC	N9	GND
H4	NC	J11	GND	K18	H_Data[50]	M3	AD[02]	N10	GND
H5	P33V	J12	GND	K19	H_Data[39]	M4	AD[00]	N11	GND

(Sheet 2 of 4)

Table 4-3
SharcFIN Pinout (Continued)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
N12	GND	P19	H_Addr[09]	T4	AD[59]	U11	H3_Flag1	V18	PR_Addr[10]
N13	GND	P20	PR_Data[4]	T5	GND	U12	H_Page	V19	H_Addr[18]
N14	GND	P21	$\overline{\text{H2\_IRQ1}}$	T6	AD[62]	U13	$\overline{\text{H\_BMS}}$	V20	PR_Addr[14]
N15	GND	P22	PR_Data[0]	T7	GND	U14	WD_TDly	V21	H_Addr[14]
N16	H_Data[61]	R1	$\overline{\text{C/BE[7]}}$	T8	GND	U15	GND	V22	H_Addr[10]
N17	H_Data[33]	R2	NC	T9	GND	U16	H_Addr[27]	W1	AD[54]
N18	PR_Data[1]	R3	$\overline{\text{C/BE[5]}}$	T10	H_Addr[02]	U17	GND	W2	AD[51]
N19	NC	R4	NC	T11	GND	U18	P33V	W3	AD[50]
N20	H_Data[32]	R5	AD[63]	T12	H3_IRQ0*	U19	PR_Addr[15]	W4	NC
N21	H_Data[34]	R6	PAR64	T13	H1_Flag1	U20	H_Addr[15]	W5	NC
N22	PR_Data[2]	R7	NC	T14	PR_Addr[04]	U21	P_SCL	W6	AD[44]
P1	AD[01]	R8	GND	T15	PR_Addr[11]	U22	P_SDA	W7	NC
P2	AD[03]	R9	P33V	T16	GND	V1	AD[55]	W8	NC
P3	NC	R10	H_EEPWP	T17	H_Addr[16]	V2	AD[56]	W9	$\overline{\text{PA\_BusM2}}$
P4	$\overline{\text{C/BE[6]}}$	R11	NC	T18	PR_Addr[16]	V3	NC	W10	H_SCL
P5	GND	R12	GND	T19	NC	V4	AD[47]	W11	PR_Addr[02]
P6	P33V	R13	$\overline{\text{H4\_DMAR}}$	T20	H_Addr[19]	V5	P33V	W12	$\overline{\text{H2\_IRQ0}}$
P7	GND	R14	P33V	T21	H_Addr[22]	V6	AD[40]	W13	$\overline{\text{H\_MS3}}$
P8	GND	R15	GND	T22	H_Addr[12]	V7	AD[36]	W14	SD_CKE
P9	GND	R16	P33V	U1	AD[61]	V8	NC	W15	H3_Flag0
P10	GND	R17	P33V	U2	AD[58]	V9	H_SDA	W16	H_Addr[26]
P11	GND	R18	H_Addr[20]	U3	AD[53]	V10	H_Addr[03]	W17	NC
P12	GND	R19	PR_ADDR[13]	U4	NC	V11	V(I/O)	W18	NC
P13	GND	R20	PR_ADDR[09]	U5	AD[57]	V12	$\overline{\text{H\_MS1}}$	W19	NC
P14	GND	R21	H_Addr[13]	U6	GND	V13	UART_Rst	W20	H_Addr[21]
P15	GND	R22	PR_Addr[19]	U7	P33V	V14	SD_A10	W21	H_Addr[11]
P16	GND	T1	$\overline{\text{C/BE[4]}}$	U8	P33V	V15	H_Addr[30]	W22	PR_Addr[12]
P17	PR_Data[3]	T2	AD[60]	U9	$\overline{\text{PA\_BusM4}}$	V16	H_Addr[31]	Y1	AD[52]
P18	PR_Addr[20]	T3	NC	U10	PR_Addr[03]	V17	P33V	Y2	AD[49]

(Sheet 3 of 4)

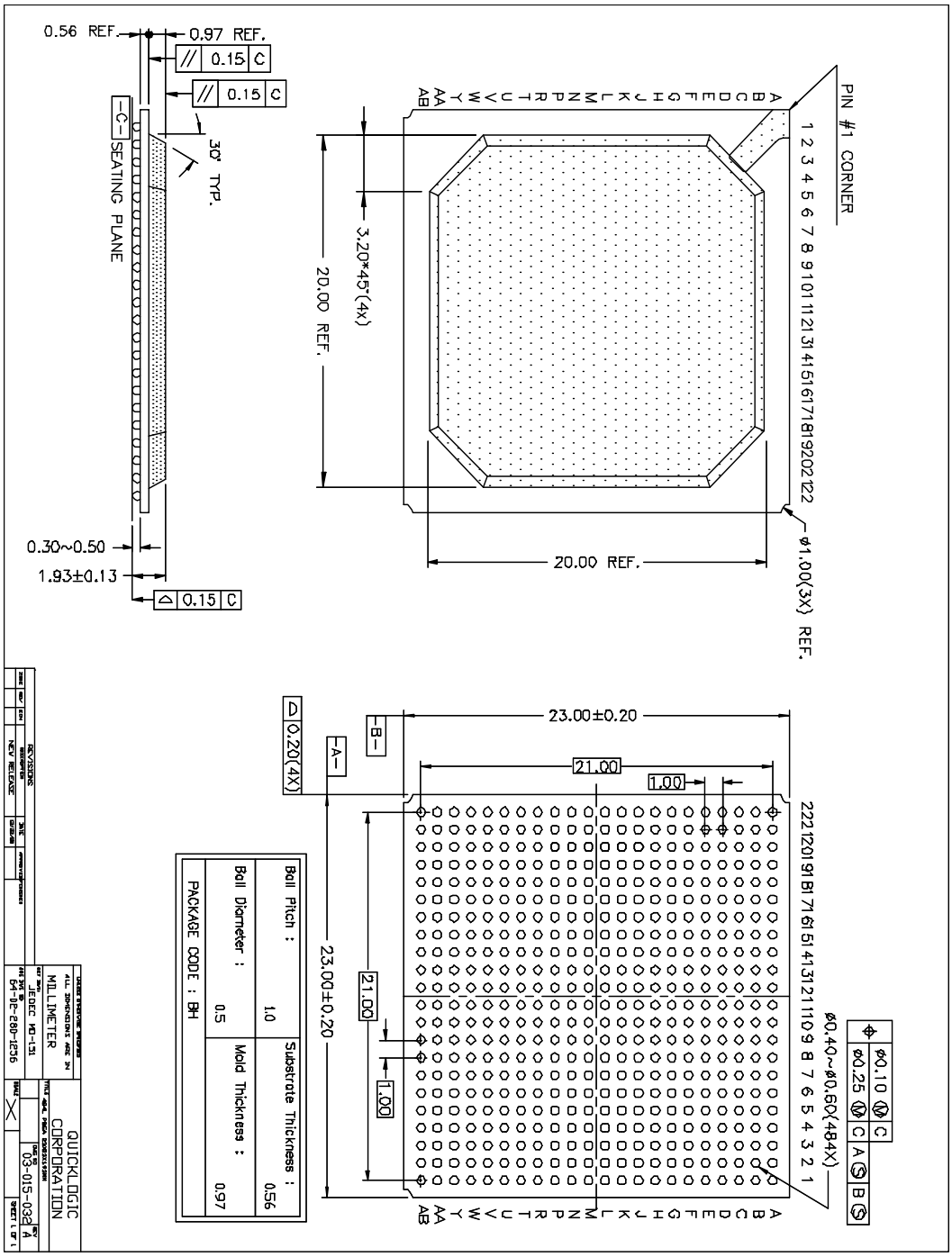
Table 4-3
SharcFIN Pinout (Continued)

Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal	Pin	Signal
Y3	NC	Y17	H_Addr[28]	AA9	NC	AB1	NC	AB15	$\overline{\text{PR\_Rst}}$
Y4	NC	Y18	H_Addr[04]	AA10	$\overline{\text{PA\_BusM3}}$	AB2	NC	AB16	H2_Flag1
Y5	AD[41]	Y19	H_Addr[25]	AA11	PR_Addr[00]	AB3	TDI	AB17	H_Addr[29]
Y6	AD[37]	Y20	NC	AA12	$\overline{\text{H4\_IRQ0}}$	AB4	AD[43]	AB18	H2_Flag0
Y7	AD[38]	Y21	H_Addr[24]	AA13	$\overline{\text{H3\_DMAR}}$	AB5	AD[46]	AB19	$\overline{\text{TRST}}$
Y8	AD[34]	Y22	H_Addr[17]	AA14	$\overline{\text{PR\_UARTSel}}$	AB6	AD[42]	AB20	PR_Addr[17]
Y9	NC	AA1	NC	AA15	$\overline{\text{SD\_CS0}}$	AB7	AD[33]	AB21	NC
Y10	$\overline{\text{H4\_IRQ1}}$	AA2	NC	AA16	PR_Addr[18]	AB8	AD[32]	AB22	NC
Y11	$\overline{\text{H3\_IRQ1}}$	AA3	AD[45]	AA17	H1_Flag0	AB9	V(I/O)	AB17	H_Addr[29]
Y12	$\overline{\text{H1\_IRQ1}}$	AA4	AD[39]	AA18	H4_Flag0	AB10	$\overline{\text{PA\_BusM1}}$	AB18	H2_Flag0
Y13	$\overline{\text{H\_MS2}}$	AA5	AD[35]	AA19	TMS	AB11	H_Addr[00]	AB19	$\overline{\text{TRST}}$
Y14	$\overline{\text{PR\_FLASHSel}}$	AA6	AD[48]	AA20	H_Addr[23]	AB12	H4_Flag1	AB20	PR_Addr[17]
Y15	$\overline{\text{SD\_CS1}}$	AA7	NC	AA21	NC	AB13	$\overline{\text{H1\_IRQ0}}$	AB21	NC
Y16	$\overline{\text{PR\_UBSel}}$	AA8	NC	AA22	NC	AB14	$\overline{\text{PR\_Int}}$	AB22	NC

(Sheet 4 of 4)

4.3 PWB Layout

Figure 4-1
484 PBGA Mechanical Drawing of the SharcFIN



Appendix A
Timing Diagrams

This appendix provides timing diagrams for the SharcFIN.

TBD