TMS320C6000 DSP Peripheral Component Interconnect (PCI) Reference Guide

Literature Number: SPRU581C
January 2007

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

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

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
Low Power Wireless	www.ti.com/lpw	Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This document describes the peripheral component interconnect (PCI) port in the digital signal processors (DSPs) of the TMS320C6000™ DSP family. Also refer to the PCI Specification revision 2.2 for details on PCI interface.

Notational Conventions

This document uses the following conventions.

- Hexadecimal numbers are shown with the suffix h. For example, the following number is 40 hexadecimal (decimal 64): 40h.
- Registers in this document are shown in figures and described in tables.
  - Each register figure shows a rectangle divided into fields that represent the fields of the register. Each field is labeled with its bit name, its beginning and ending bit numbers above, and its read/write properties below. A legend explains the notation used for the properties.
  - Reserved bits in a register figure designate a bit that is used for future device expansion. Reads from this reserved bit always return the default value. Writes to this reserved bit are ignored. If writing to this field, always write the default value for future device compatibility.

31	1	0
Reserved†		Bit
R-0		R/W-0

Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Related Documentation From Texas Instruments

The following documents describe the C6000™ devices and related support tools. Copies of these documents are available on the Internet at www.ti.com. *Tip:* Enter the literature number in the search box provided at www.ti.com.

TMS320C6000 CPU and Instruction Set Reference Guide (literature number SPRU189) describes the TMS320C6000™ CPU architecture, instruction set, pipeline, and interrupts for these digital signal processors.

TMS320C6000 DSP Peripherals Overview Reference Guide (literature number SPRU190) describes the peripherals available on the TMS320C6000™ DSPs.

TMS320C6000 Technical Brief (literature number SPRU197) gives an introduction to the TMS320C62x™ and TMS320C67x™ DSPs, development tools, and third-party support.

TMS320C64x Technical Overview (SPRU395) gives an introduction to the TMS320C64x™ DSP and discusses the application areas that are enhanced by the TMS320C64x VelociTI™.

TMS320C6000 Programmer's Guide (literature number SPRU198) describes ways to optimize C and assembly code for the TMS320C6000™ DSPs and includes application program examples.

TMS320C6000 Code Composer Studio Tutorial (literature number SPRU301) introduces the Code Composer Studio™ integrated development environment and software tools.

Code Composer Studio Application Programming Interface Reference Guide (literature number SPRU321) describes the Code Composer Studio™ application programming interface (API), which allows you to program custom plug-ins for Code Composer.

TMS320C6x Peripheral Support Library Programmer's Reference (literature number SPRU273) describes the contents of the TMS320C6000™ peripheral support library of functions and macros. It lists functions and macros both by header file and alphabetically, provides a complete description of each, and gives code examples to show how they are used.

TMS320C6000 Chip Support Library API Reference Guide (literature number SPRU401) describes a set of application programming interfaces (APIs) used to configure and control the on-chip peripherals.

Trademarks

Code Composer Studio, C6000, C62x, C64x, C67x, TMS320C6000, TMS320C62x, TMS320C64x, TMS320C67x, and VelociTI are trademarks of Texas Instruments.

Contents

Figures

Tables

Peripheral Component Interconnect (PCI)

This manual describes the peripheral component interconnect (PCI) port in the digital signal processors (DSPs) of the TMS320C6000™ DSP family. Also refer to the PCI Specification revision 2.2 for details on PCI interface.

1 Overview

The PCI port supports the following PCI features:

- ❑ Conforms to PCI specification revision 2.2
- ❑ Conforms to power management interface specification revision1.1 (C6205 DSP only)
- ❑ Meets requirements of PC99
- ❑ PCI master/slave interface
- ❑ 32-bit address/data bus
- ❑ Single function device
- ❑ Medium address decode
- ❑ PCI access to all on-chip RAM, peripherals, and external memory via external memory interface (EMIF)
- ❑ Supports memory read, memory read multiple, memory read line, memory write commands, I/O read, I/O write, CFG read, and CFG write.
- ❑ Unlimited slave-access burst lengths
- ❑ Master transfers of up to 64K bytes
- ❑ Single-word transfers for I/O read/writes
- ❑ Single-word transfers for configuration register access
- ❑ Many configuration register contents (subsystem ID/subsystem vendor ID, etc.) initialized from an external serial EEPROM at PCI reset.
- ❑ Supports 4-wire serial EEPROM interface
- ❑ EEPROM interface used directly by PCI port without DSP intervention on PCI reset. DSP software control of EEPROM after PCI reset.

- PCI interrupt request under DSP program control
- DSP interrupt via PCI I/O cycle
- DSP power control via software (C6205 DSP only)
- Peripheral power control via software (C6205 DSP only)
- Software-controlled assertion of PME from D0, D1, D2, $D3_{hot}$ (C6205 DSP only)
- Hardware-controlled assertion of PME on power wakeup active from $D3_{cold}$. Optional hardware-controlled assertion of PME from D0, D1, D2, $D3_{hot}$. (C6205 DSP only)
- Supports D0, D1, D2, $D3_{hot}$, $D3_{cold}$ power management modes (C6205 DSP only)
- Implements PCI power management control status register “sticky” bits from logic powered by $3.3V_{aux}$ (C6205 DSP only)
- Four FIFOs for efficient data transfer (master write, master read, slave write, slave read)
- Independent master/slave operation
- Independent slave read/slave write operation
- Three PCI base address registers (prefetchable memory, nonprefetchable memory, I/O)
- Disconnect with retry on memory read line, memory read multiple to prefetchable memory
- No wait states inserted by DSP on PCI master or slave transactions

The PCI port does not support:

- PCI special cycles
- PCI interrupt acknowledge cycles
- PCI lock
- PCI memory caching
- 64-bit bus operation
- Master address/data stepping
- Master combining (for write posting)
- Collapsing
- Merging
- Cache line-wrap accesses
- Reserved accesses
- Message-signaling interrupts
- Vital product data
- Compact PCI hot swap

The PCI port supports connection of the DSP to a PCI host via the integrated PCI master/slave bus interface. For C62x devices, the PCI port interfaces to the DSP via the auxiliary channel of the DMA controller (Figure 1). For C64x devices, the PCI port interfaces to the DSP via the enhanced DMA (EDMA) controller (Figure 2). This architecture allows for both PCI master and slave transactions, while keeping the DMA/EDMA channel resources available for other applications.

The C62x PCI port provides the auxiliary DMA with a source/destination address in the DSP memory. Address decode is performed by the DMA to select the appropriate interface (data memory, program memory, register I/O, or external memory). The auxiliary channel of the DMA controller should be programmed for the highest priority in order to achieve the maximum throughput on the PCI interface.

The C64x PCI port uses the EDMA internal address generation hardware to perform address decode instead.

Table 1 lists the differences between the C62x and C64x PCI ports.

Figure 1. TMS320C62x DSP Block Diagram

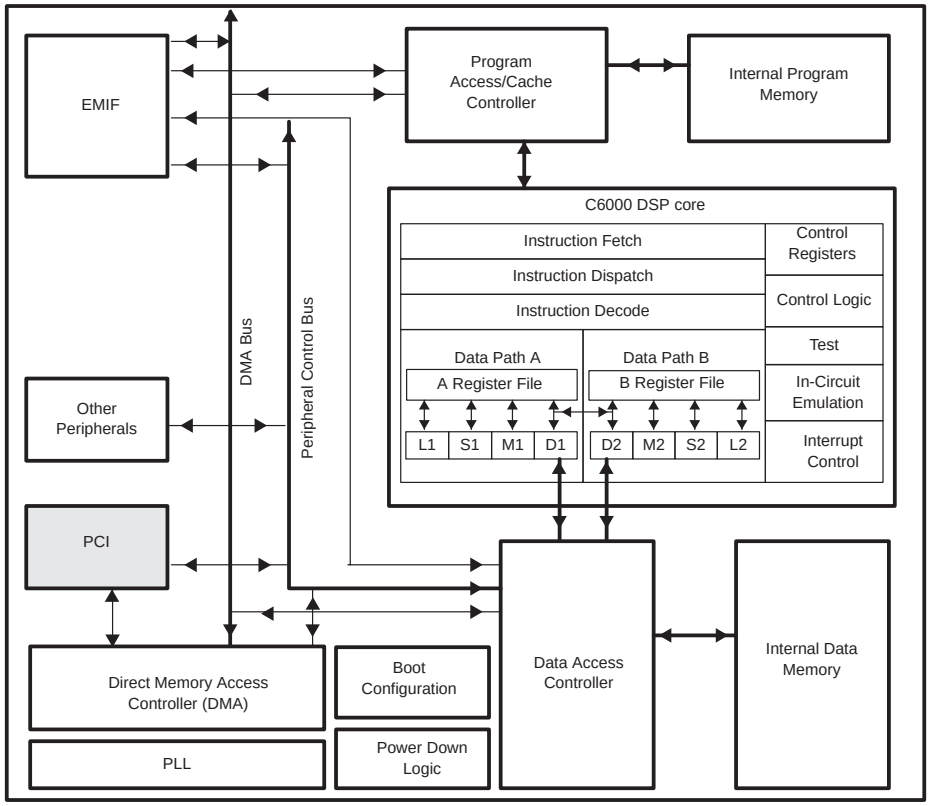


Figure 2. TMS320C64x DSP Block Diagram

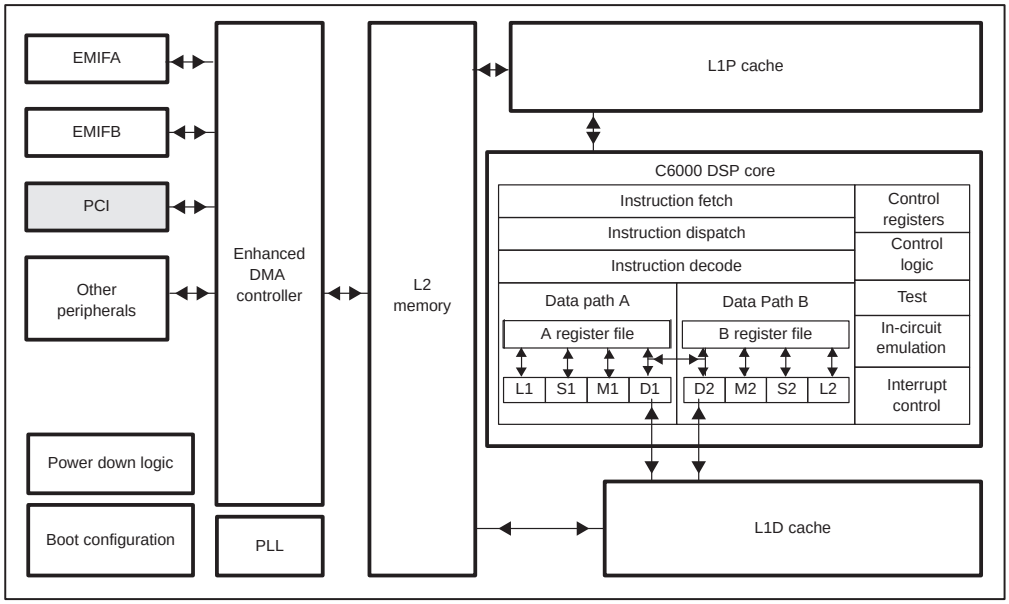


Table 1. Differences Between the C62x and C64x PCI

Features	C62x PCI	C64x PCI
Internal transfer	Auxiliary DMA	EDMA internal address generation hardware
PMDCSR register	Supported	Does not apply
HALT register	Supported	Does not apply
TRCTL register	Does not apply	Supported
Supported EEPROM size, bits	1K, 2K, 4K, 16K	4K
Power management support	Yes	No
FIFO depth	8 words	16 words

† P = CPU clock period

2 PCI Architecture

The PCI port supports four types of PCI data transactions:

- Slave Writes: External PCI master writes to DSP slave
- Slave Reads: External PCI master reads from DSP slave
- Master Writes: DSP master writes to external slave
- Master Reads: DSP master reads from external slave

The PCI port block diagram, shown in Figure 3 (page 19), consists of the following primary blocks:

- PCI Bus Interface Unit (PBIN)

  The PCI bus protocol is implemented in the PCI bus interface unit (PBIN). To maximize PCI bus bandwidth, the PCI interface does not insert wait states for slave or master burst transactions. If the corresponding FIFO goes full or empty, the PCI interface disconnects the current transfer. The PCI bus interface unit inserts the following delays on the PCI bus:

  - Slave Writes**:**
    - Zero wait state initial transfer
    - Zero wait state subsequent transfers
    - Disconnect if FIFO is full, or previous frame is not complete
  - Slave Reads:
    - Prefetchable: Initial access disconnected with retry
    - Up to 16 wait states inserted for single-word transfers
    - Zero wait state initial transfer (prefetchable retry)
    - Zero wait state subsequent transfers (prefetchable retry)
    - Disconnect if FIFO is empty, or other slave read frame is in progress
  - Master Writes:
    - Zero wait state initial transfer
    - Zero wait state subsequent transfers
  - Master Reads:
    - Zero wait state initial transfer
    - Zero wait state subsequent transfers

- EEPROM Controller

  The EEPROM controller interfaces to the 4-wire serial EEPROM interface. On PCI reset, the EEPROM controller reads the EEPROM and provides the PCI bus interface unit with the configuration data. During normal operation, the EEPROM can be accessed by the DSP via the memory-mapped registers.

- DSP Slave Write Block

  The DSP slave write block contains a multiplexer and a FIFO to transfer data (written by the external PCI master) from the PCI bus interface unit to the DSP.

- DSP Slave Read Block

  The DSP slave read block contains a multiplexer and a FIFO to transfer data from the DSP to the PCI bus interface unit. The external PCI master is the requester of this data.

- DSP Master Block

  The DSP master block is divided into the read and write portions. The device cannot perform both read and write operations simultaneously.

  - The write portion of the DSP master block contains a data multiplexer and a FIFO for DSP master writes. It transfers data from the DSP (master) to an external slave via the PCI bus interface unit.
  - The read portion of the DSP master block contains a data multiplexer and a FIFO for DSP master reads. It transfers data from the PCI bus interface unit to the DSP (master).

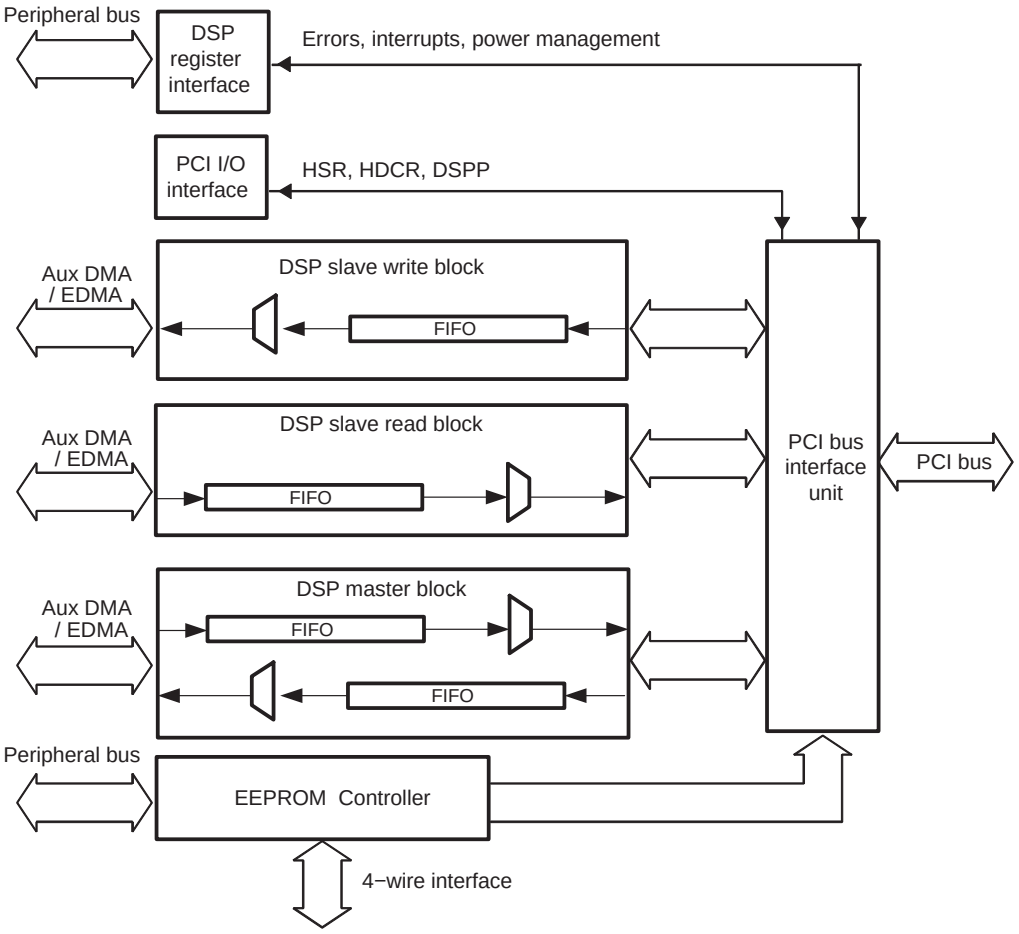
- PCI I/O Interface

  The input/output (I/O) interface contains the PCI I/O-mapped registers. These registers control the DMA/EDMA page for slave transactions, indicate the host status, and can interrupt or reset the DSP core.

- DSP Register Interface

  The DSP register interface contains DSP memory-mapped registers for the control of the master interface, PCI host interrupts, and power management.

Figure 3. PCI Port Block Diagram



3 Memory Map

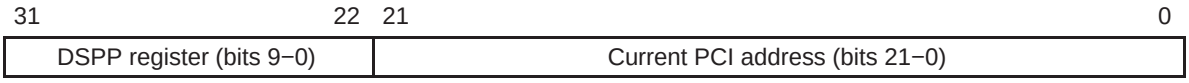
The PCI port has full visibility into the DSP memory map through three base address registers:

- Base 0: 4M-byte prefetchable maps to all of DSP memory with the DSP page register (DSPP). Prefetch reads have all bytes valid.
- Base 1: 8M-byte nonprefetchable maps to DSP memory-mapped registers. Nonprefetch supports byte enables.
- Base 2: 16-byte I/O contains I/O registers for the PCI host

These three registers belong to the group of PCI configuration registers. PCI host accesses to DSP (prefetchable) memory are mapped to a 4M-byte window in the PCI memory space. The PCI port contains a PCI I/O register, the DSP page register (DSPP), that specifies the address mapping from the PCI address to the DSP address. This address mapping is used when the DSP is a slave on the PCI local bus.

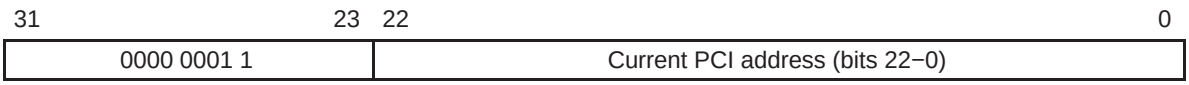
The DSPP, described in section 15.2.3, is used to locate the 4M-byte window within the DSP memory map. Bits 21−0 of the PCI address are concatenated with bits 9−0 of DSPP to form the DSP address for PCI slave access to the DSP as shown in Figure 4.

Figure 4. PCI Base Slave Address Generation (Prefetchable)



The PCI base 1 register on the DSP is configured for an 8M-byte nonprefetchable region. This memory is mapped into the DSP at a fixed location (0180 0000h−0200 0000h). Bits 22−0 of the PCI address are concatenated with a fixed offset to map the base 1 access into the memory-mapped registers as shown in Figure 5.

Figure 5. PCI Base 1 Slave Address Generation (Nonprefetchable)



Base address register 2 is configured for a 16-byte I/O region for the PCI host to access the PCI I/O registers. See section 15.2.

The PCI bus interface provides two access methods for PCI host access to DSP memory. The 4M-byte base 0 region is used for prefetchable data, and the 8M-byte base 1 region is used for nonprefetchable (register) access. All transfers to the nonprefetchable region transfer single words and then disconnect.

Data access to the prefetchable region may be transferred in bursts limited mainly by the host system setup (PCI bridge latency timer, burst length count). Prior to transferring data, the PCI host must first write DSPP to locate the 4M-byte window within the DSP memory map.

PCI master transactions issued by the DSP will attempt to use bursts. Through disconnects, however, the external slave can force the DSP master to perform single-word transfers.

Internal to the DSP, all data transfers are handled by the auxiliary channel of the DMA controller (C62x devices) or the EDMA internal address generation hardware (C64x devices).

Note: You must ensure that no PCI transactions cross the port boundaries of the DMA/EDMA controller. A port boundary is the address boundary between external memory and internal memory, between external memory and the peripheral address space, or between internal memory and the peripheral address space.

4 Byte Addressing

The PCI interface is byte-addressable. The PCI interface can read and write 8-bit bytes, 16-bit halfwords, 24-bit words, and 32-bit words. Words are aligned on an even four-byte boundary. Words always start at a byte address where the two LSBs are 00. Halfwords always start at a byte address where the last LSB is 0.

PCI slave transactions to non-prefetchable spaces are fully byte-addressable. PCI slave transactions to prefetchable spaces are done with no smaller than word granularity.

5 PCI Address Decode

The PCI port supports “medium” address decode of the PCI address for memory and I/O cycles. The $\overline{\text{PDEVSEL}}$ signal is asserted two PCI clock periods after $\overline{\text{PFRAME}}$ is sampled and asserted.

6 Special Considerations for PCI Memory Service

6.1 Prefetchable Reads (C62x DSP only)

When the PCI executes a read from the DSP memory space, it performs burst prefetches of four words. This results in the DMA auxiliary channel reading three higher-word addresses that you may not have explicitly requested. This occurs only when he PCI port performs prefetchable reads on behalf of an external PCI master.

These prefetchable reads can generate the following undesired operations:

1) Accesses to undesired CE spaces:
   - When reading the top three words of EMIF CE0, the resulting prefetches can cause an inadvertent access to CE1 that may cause an undesired read to a device or a stall if the inadvertent access is to an asynchronous memory space with ARDY left floating or pulled inactive (not-ready).
   - The above example also applies to CE2 with the resulting prefetches possibly causing an inadvertent access to CE3.

   Associated design tip: If not using ARDY, always pull to the ready state to avoid stalls. If you always want to detect bad software setups, always pull to the not-ready state to detect system stalls.

2) Unintended port crossings or illegal accesses to a reserved location:
   - When reading the top three words of EMIF CE1, the access can cross into either program memory (PMEM) block 0 when in map 0 or to the internal peripheral bus (PBus) region storing EMIF control registers when in map 1. This is an illegal port crossing.
   - When reading the top three words of EMIF CE3, the access can cross into reserved address space. This is an illegal access.
   - When reading the top three words of PMEM block 0 the access can cross into PMEM block 1. This is an illegal port crossing.
   - When reading the top three words of PMEM block 1, the access can cross into reserved address space. This is an illegal access.
   - When reading the top three words of data memory (DMEM) block 1, the access can cross into reserved address space. This is an illegal access.
   - When reading anywhere in the PBus space, you may prefetch ahead to three undesired control registers. This can cause an illegal access when accessing a reserved register address. If the register access has side effects (like reading the McBSP DRR, clearing RRDY), then you may inadvertently cause these side effects.

Note: A restriction does not exist when crossing between DMEM block 0 and block 1 because they both use the same DMA port.

Associated design tips:

- When reading internal peripheral registers:
  - For reads from an external master, use fixed-mode addressing. As a broader statement, it is good practice to use fixed mode also when writing to peripheral registers as sometimes there are gaps between them.
  - On PCI, always use non-prefetchable reads of peripheral registers.
- When reading the top three locations of an EMIF CEx, internal program block or DMEM block 1, use fixed-mode addressing. Note this procedure does not have to be followed when accessing the top three words of DMEM block 0, this is because DMEM block 0 and block 1 are in the same DMA port.

6.2 PCI Transfers To and From Program Memory (C62x DSP only)

The CPU has priority over the DMA (and auxiliary DMA) for access to program memory. Since the CPU can access the program memory on every CPU clock, it can possibly lock out the DMA from accessing the program memory.

If the PCI port is requesting program memory transfers via the auxiliary channel, all the other four DMA channels are halted. Thus, no DMA channel activity occurs during the PCI port requests, even if a DMA channel is accessing a different memory (EMIF/peripheral/data memory). If the CPU has higher priority, all DMA activity can be blocked while the CPU is executing a tightly coded routine from program memory.

The PCI transfer halt register (HALT) prevents the PCI port from performing master/slave auxiliary channel requests. If the HALT bit is set, all auxiliary transfers are prevented. Any current PCI master transaction will complete its DMA cycle. The PCI transaction will not commence until the HALT bit is deasserted. This prevents DMA lockup when a PCI transaction is in progress and the DSP is executing a packed section of code. The other DMA channels are free to access memory when the PCI master is halted.

The HALT register does not apply to C64x devices, because the DMA lockup condition does not apply. The C64x EDMA uses a priority queue implementation. Only EDMA transfers placed in the same priority queue as the PCI transfer will be stalled in the above condition. See *TMS320C6000 DSP Enhanced DMA (EDMA) Controller Reference Guide* (SPRU234), for details on priority queues.

6.3 PCI EDMA Usage Control (C64x DSP only)

The PCI port uses the EDMA hardware for memory requests. Use of this shared resource is governed by the PCI transfer request control register (TRCTL), and is user-configurable, provided care is taken when changing the values in the register.

To safely change the PALLOC or PRI bits in TRCTL, the TRSTALL bit needs to be used to ensure a proper transition. The following procedure must be followed to change the PALLOC or PRI bits:

1) Set the TRSTALL bit to 1 to stop the PCI from submitting TR requests on the current PRI level. In the same write, the desired new PALLOC and PRI bits may be specified.
2) Clear all EDMA event enables (EER) corresponding to both old and new PRI levels to stop the EDMA from submitting TR requests on both PRI levels. Do not manually submit additional events via the EDMA.
3) Do not submit new QDMA requests on either old or new PRI level.
4) Stop L2 cache misses on either old or new PRI level. This can be done by forcing program execution or data accesses in internal memory. Another way is to have the CPU executing a tight loop that does not cause additional cache misses.
5) Poll the appropriate PQ bits in the priority queue status register (PQSR) of the EDMA until both queues are empty (see *TMS320C6000 DSP Enhanced DMA (EDMA) Controller Reference Guide*, SPRU234).
6) Clear the TRSTALL bit to 0 to allow the PCI to continue normal operation.

Requestors are halted on the old PCI PRI level so that memory ordering can be preserved. In this case, all pending requests corresponding to the old PRI level must be allowed to complete before PCI is released from stall state.

Requestors are halted on the new PRI level to ensure that at no time can the sum of all requestor allocations exceed the queue length. By halting all requestors at a given level, you can be free to modify the queue allocation counters of each requestor.

7 Slave Transfers

7.1 DSP Slave Writes

The slave write FIFO in the DSP slave write block (Figure 3, page 19) is used to efficiently handle PCI host writes to the DSP slave. The address for a DSP slave write is derived from the PCI address concatenated with the fixed offset in DSPP, as described in section 3. No wait states are inserted by the slave PCI port. DSP slave writes execute with zero wait states on all data phases for both single and burst accesses. The PCI interface supports unlimited length memory burst transfers.

Slave write access to DSP is only disconnected when the FIFO is full, or when the FIFO is not empty from a previous PCI slave write frame. Slave reads and master read/write transactions have no effect on slave write PCI transactions.

Internally, the auxiliary DMA or EDMA internal address generation hardware service the slave write FIFO when:

- ❏ FIFO has at least 4 words of data, or
- ❏ PCI transaction has terminated ($\overline{\text{PFRAME}}$ deasserted)

The DSP slave write address is autoincremented internally. DSP memory writes continue until there is no longer any valid data in the FIFO. This applies to both single and burst PCI transactions. For single access transactions, the internal transfer request is made after the PCI transaction has terminated.

7.2 DSP Slave Reads

Similar to slave writes, the slave read FIFO in the DSP slave read block (Figure 3, page 19) is used to efficiently handle PCI host reads from the DSP slave. The PCI slave read interface supports unlimited length memory burst transfers.

The PCI port uses the cache line size and PCI command to determine the number of bytes transferred for a slave read. The type of PCI access is indicated by the PCI command/byte enable signals ($\overline{\text{PCBEx}}$) during the address phase. The following slave read commands are supported:

- ❏ Memory read
- ❏ Memory read multiple
- ❏ Memory read line

All of the above PCI slave reads can be prefetchable and nonprefetchable.

7.2.1 Nonprefetchable Slave Reads (BAR 1)

For nonprefetchable slave reads, the PCI port inserts wait cycles until the requested word, halfword, or byte is written to the FIFO. The data is then transferred on the PCI bus, and the cycle is terminated regardless if the command was a memory read, a memory read multiple, or a memory read line.

7.2.2 Prefetchable Slave Reads (BAR 0)

For prefetchable slave reads, the PCI port inserts wait cycles until the requested word is ready. The PCI port adheres to the 16-clock rule, if data is not ready in 16 PCI clocks, the memory read is disconnected with retry. Memory read commands always produce one (1) word of data. Memory Read Line and Memory Read Multiple should be used for burst accesses.

7.2.3 Prefetchable Slave Read Multiple and Prefetchable Slave Read Line (BAR 0)

These requests are initially terminated on the PCI bus with a disconnect with retry. Subsequently, the auxiliary DMA or EDMA internal address generation hardware services the PCI port by transferring read data to the FIFO. The PCI slave transfer occurs when the original master reattempts the initial transfer. Nonrelated slave requests are terminated with a disconnect.

For memory read line commands, the number of bytes transferred to the slave read FIFO is based on the cache-line size register. For memory read multiple commands, the DSP slave read continuously fills the FIFO with data until the PCI master terminates the transaction ($\overline{\text{PFRAME}}$ deasserted), at which point the last PCI valid data sample is transferred and the FIFO is flushed.

The auxiliary DMA or EDMA internal address generation hardware bursts until the slave read FIFO is full. The DSP slave address is autoincremented internally.

Memory read line and memory read multiple commands transfer data with zero wait states inserted by the PCI port, when the requestor retries the command and the FIFO has data. The FIFO request is also terminated, if the PCI transaction is disconnected prematurely by the master.

7.3 PCI Target-Initiated Termination

The DSP issues target terminations under these conditions:

- Data transfer with disconnect, if the master issues a burst memory access with an addressing mode that is not supported.
- Retry with data transfer, if the master attempts a burst access to the configuration space (see section 15.1).
- Retry with data transfer, if the master attempts a burst access to the I/O space (see section 15.2).
- Disconnect for slave memory or I/O writes and a transaction is waiting in the internal slave read FIFO.
- Disconnect for slave memory reads, if the PCI address value does not match the address in the internal read prefetch buffer.
- Once a prefetch has been started, retry for all other memory and I/O reads until the original posted transaction is repeated by the PCI bus master and the prefetched data is transferred.

The PCI interface meets all 16-clock and 8-clock rules for data transfers within single access and burst accesses.

8 Master Transfers

Master transfers are initiated under DSP control. The following PCI memory-mapped peripheral registers are used to configure a DSP master transfer:

- DSP master address register (DSPMA)
- PCI master address register (PCIMA)
- PCI master control register (PCIMC)

The following PCI memory-mapped peripheral registers indicate the status of the current master transfer:

- Current DSP address register (CDSPA)
- Current PCI address register (CPCIA)
- Current byte count register (CCNT)

For C62x devices, the PCI transfer halt register (HALT) allows the DSP to terminate internal transfer requests to the auxiliary DMA channel.

8.1 DSP Master Writes

The master write FIFO in the DSP master block (Figure 3, page 19) is used to efficiently handle DSP master writes to an external slave. The master write interface supports burst lengths of up to 64K − 1 bytes.

Master writes are initiated under DSP control via the DSP master address register (DSPMA), the PCI master address register (PCIMA), and the PCI master control register (PCIMC).

For DSP master writes, the ADDRMA bits in DSPMA contain the word-aligned source (DSP) address. For the C6205 DSP only, if the AINC bit in DSPMA is cleared, the source address is autoincremented by 4 bytes after each internal data transfer. On the C64x, all transfers autoincrement. Note that for both C6205 and C64x, DSPMA will never autoincrement past a 16MB boundary. Instead, it will wrap around within the 16MB block. PCIMA contains the word-aligned destination (PCI) address. An internal register keeps track of the PCI master address.

A master write is initiated by enabling the START bits in PCIMC. The auxiliary DMA or EDMA transfers data from the source address (pointed to by DSPMA) to the master write FIFO. It either fills up the FIFO or transfers only the number of words desired if that number is less than the FIFO length. Subsequent internal data transfers are performed when the FIFO has space for 4 or more words of data. Internal data transfer continues until the FIFO is full or until the transfer is completed.

Once the FIFO has valid data, a PCI bus request is made and data is transferred from the FIFO to the PCI. DSP master writes execute with zero wait states on all data phases of both single and burst accesses. The PCI command/byte enable signals ($\overline{\text{PCBEx}}$) indicate the master write bytes on the PCI interface.

Internal data transfers stop once all master write data has been transferred from the DSP source to the master write FIFO. For C62x devices, internal data transfers and the current PCI bus cycle can also be terminated by asserting the HALT bit in the HALT register. The PCI bus interface monitors the PCI interface for disconnects, retries, and target-aborts. The PCI port complies with PCI Specification revision 2.2 and retries the exact same cycle during retries.

If the cycle is terminated with a master abort or a target abort, the current transfer is terminated both internally and externally on the PCI bus. The master write FIFO is flushed and either the master abort (PCIMASTER) or target abort (PCITARGET) bit is set in the PCI interrupt source register (PCIIS). These error conditions can generate a CPU interrupt, if the corresponding bits are set in the PCI interrupt enable register (PCIIEN).

If the PCI latency timer (specified in the PCI configuration register space) times out, the PCI master gives up the bus. The master requests the bus later and completes the necessary transfers.

The progress of the transfer can be polled by reading the PCI master control register (PCIMC). The START bits go to 000b, when the transfer is complete on both the DSP and PCI side. Alternatively, the master can be programmed to generate an interrupt upon completion of a frame transfer by setting the MASTEROK bit in PCIIEN.

8.2 DSP Master Reads

The master read FIFO in the DSP master block (Figure 3, page 19) is used to efficiently handle DSP master reads from an external slave. The master read interface supports burst lengths of up to 64K − 1 bytes.

Master reads are initiated under DSP control via the DSP master address register (DSPMA), the PCI master address register (PCIMA), and the PCI master control register (PCIMC).

For DSP master reads, PCIMA contains the external PCI slave source address. The ADDRMA bits in DSPMA contain the word-aligned destination (DSP) address. For the C6205 DSP only, if the AINC bit in DSPMA is cleared, the destination address is autoincremented by 4 bytes after each internal data transfer. On the C64x, all transfers autoincrement. Note that for both C6205 and C64x, DSPMA will never autoincrement past a 16MB boundary. Instead, it will wrap around within the 16MB block.

A master memory read is initiated by enabling the START bits in PCIMC. The PCI port performs a PCI bus request. Once a PCI bus request is granted, a PCI bus cycle is initiated. The type of cycle initiated depends on the number of bytes to be transferred and the cache line size. The following master memory read commands are supported:

- Memory read
- Memory read multiple
- Memory read line

You can initiate two types of memory reads, based on the START bits in PCIMC. Prefetchable memory reads (START = 010b) use the memory read multiple and memory read line commands for transfers greater than one word. A memory read command is used for transfers of one word.

Nonprefetchable memory reads (START = 011b) always use a memory read command. A transfer size of *N* words is broken up into *N* one-word read cycles on the PCI bus. You should read from prefetchable memory whenever possible.

No wait states are inserted on the initial data phase or subsequent data phases of the PCI master read access. Read data is written to the master read FIFO. An internal auxiliary DMA or EDMA data transfer request occurs when the FIFO has at least 4 words of data, or when the PCI transaction has terminated. The auxiliary DMA channel or EDMA transfers data from the master read FIFO to the DSP destination address (ADDRMA bits in DSPMA). All master read transactions are word-aligned.

Auxiliary DMA or EDMA transfers continue until there are no longer any valid data in the FIFO. This applies to both single access and burst PCI transactions. For single access transactions, the internal data transfer occurs after the PCI transaction has terminated ($\overline{\text{PFRAME}}$ deasserted).

Internal data transfers stop once all master read data has been transferred from the master read FIFO to the DSP destination. For C62x devices, internal data transfers and the current PCI bus cycle can also be terminated by asserting the HALT bit in the HALT register. The PCI bus interface monitors the PCI interface for disconnects, retries, and target-aborts. The PCI port complies with PCI Specification revision 2.2 and retries the exact same cycle during retries.

If the cycle is terminated with a master abort or a target abort, the current transfer is terminated both internally and externally on the PCI bus. The master read FIFO is flushed and either the master abort (PCIMASTER) or target abort (PCITARGET) bit is set in PCI interrupt source register (PCIIS). These error conditions can generate a CPU interrupt, if the corresponding bits are set in the PCI interrupt enable register (PCIIEN).

If the PCI latency timer (specified in the PCI configuration register space) times out, the PCI master gives up the bus. The master requests the bus later and completes the necessary transfers.

The progress of the transfer can be polled by reading the PCI master control register (PCIMC). The start bits go to 000b when the transfer is complete on the PCI side. Alternatively, the master can be programmed to generate an interrupt upon completion of frame transfer by setting the MASTEROK bit in PCIIEN.

8.2.1 Master Read Completion

When the start bits return to 000b, or the MASTEROK interrupt is generated, the current transaction is complete from the PCI point of view. However, in the case of reads, this means the data has been handed off to DMA or EDMA for actual memory service. Due to differing paths for PCI data versus interrupts or register reads, the CPU may receive notification before data has landed. You should be careful when doing master reads to ensure that the CPU reads the correct data from memory.

This is a more likely problem in EDMA systems, due to its longer latencies. A simple way to ensure data correctness is to preform a small dummy QDMA access on the same priority queue as the PCI. When this QDMA completes, you can be sure the PCI data has landed.

8.3 DSP as System Host

The PCI port can perform master configuration cycles regardless of the master bit setting in the PCI configuration registers. Also, the internal slave and master can run concurrently, allowing the DSP to read and write its own configuration registers over the PCI bus. These capabilities allow the PCI port to act as a system host by configuring itself and the rest of the PCI bus. This is subject to the following limitations:

- ❏ IDSELs must be under DSP control.
- ❏ There is just one PCI bus segment, that is, the DSP cannot generate type 1 configuration cycles.
- ❏ External logic to arbitrate the bus.

8.3.1 Generating IDSEL

To properly configure the bus, the DSP must be able to assert the IDSEL line of each device at the appropriate time. The PCI interface does not directly provide for this; it must be accomplished by indirect means. The following sections provide two different procedures to generate IDSEL.

8.3.1.1 Resistively Coupled IDSEL

Resistively coupling IDSEL is a common and easy method of generating IDSELs; however, you must be careful to adhere to PCI specification with this method. Using coupled IDSELs requires the host to step the address phase onto the bus to allow IDSEL sufficient time to propagate to the target device. The PCI interface does not do address stepping. The compact nature of most systems where the DSP is the host should allow for a functional coupling without address stepping. Regardless, it is your responsibility to ensure the IDSEL signal propagates through the resistive coupling in time to meet PCI's timing specification.

When using resistive coupling for IDSEL generation, the procedure to read or write a configuration word is as follows:

1) Set PCIMA:
   a) PCIMA[31−11] = one bit set to select appropriate device's IDSEL (important: only one bit should be set at a time)
   b) PCIMA[10−8] = device function to be addressed
   c) PCIMA[7−0] = 32-bit configuration word address (See PCI specification for type 0 configuration access details)
2) Set DSPMA to point to a local buffer (buffer contains source data for writes, space for data for reads)
3) Set PCIMC for 4 bytes, desired function
4) Poll for START==0 or wait for MASTEROK interrupt

8.3.1.2 GPIO Generated IDSEL

An alternate and fully-compliant method of generating IDSEL is to attach the IDSEL lines to GPIO pins and control them directly. The procedure is only slightly different from above:

1) Set GPIO to enable one IDSEL only (important: only one IDSEL should be enabled at a time) You should read back new values written to GPIO pins to avoid race conditions.
2) Set PCIMA:
   a) PCIMA[11−8] = device function to be addressed
   b) PCIMA[7−0] = 32-bit configuration word address (See PCI specification for type 0 configuration access details)
3) Set DSPMA to point to a local buffer (buffer contains source data for writes, space for data for reads)
4) Set PCIMC for 4 bytes, desired function
5) Poll for START==0 or wait for MASTEROK interrupt
6) Set GPIO to turn off IDSEL

9 Reset

9.1 PCI Reset of DSP

The PCI host can reset the DSP via the host-to-DSP control register (HDCR). Setting the WARMRESET bit in HDCR to 1 causes a DSP reset, which resets all of the internal CPU and peripheral logic. WARMRESET can wake the DSP from PD2 and PD3 power-down modes. WARMRESET does not relatch the boot configuration pins.

9.2 FIFO Resets

The PCI FIFOs and control logic are held in reset when either the DSP is reset or the PCI pin $\overline{\text{PRST}}$ is asserted.

9.3 PCI Configuration Register Reset

The PCI configuration registers read from the EEPROM are initialized on the PCI bus ($\overline{\text{PRST}}$) reset. Neither the DSP core reset nor the power on reset, affects them.

9.4 PCI Behavior During DSP Reset

If the device reset is asserted, the PCI port will disconnect all incoming transactions until the device is taken out of reset. The PCI port will not perform any master transactions while the device is in reset.

10 Interrupts and Status Reporting

The PCI port can generate the following CPU interrupts:

- PCI_WAKEUP (C62x devices): this is dedicated to PCI power wake-up events.
- ADMA_HLT (C62x devices): this interrupt is generated when the auxiliary DMA is halted (DMAHALTED).
- DSPINT: this interrupt is asserted and enabled in the PCI interrupt enable register (PCIIEN), when any of these events occurs: PWRMGMT, PCITRAGET, PCIMASTER, HOSTSW, PWRLH, PWRHL, MASTEROK, CFGDONE, CFGERR, EERDY, or $\overline{\text{PRST}}$.

The PCI master/slave interface status/errors are shown in the PCI interrupt source register (PCIIS). All status/error conditions can generate a CPU interrupt, if they are enabled in the PCI interrupt enable register (PCIIEN). Status bits in PCIIS are still set, even if the interrupt is not enabled. If an enabled interrupt occurs, it sends a DSPINT interrupt to the DSP. Writing a 1 to the corresponding PCIIS bit(s) clears the interrupt. If an interrupt bit is still set, a new interrupt will then occur. Thus, upon a PCI interrupt, you should perform the following in the interrupt service routine:

- Read the PCIIS register
- Clear the appropriate PCIIS bits by writing a 1 to them

The PCI port can generate interrupts to both the CPU and the PCI host (via the $\overline{\text{PINTA}}$ pin). The following sections describe these two types of interrupts.

10.1 Host Interrupt to the DSP

The PCI host can generate an interrupt to the DSP by writing the DSPINT bit in the PCI I/O host-to-DSP control register (HDCR). Writing this bit causes the HOSTSW interrupt, if it is enabled in the PCI interrupt enable register (PCIIEN).

10.2 DSP to Host Interrupt

The DSP can generate an interrupt to the PCI host via the $\overline{\text{PINTA}}$ pin. Interrupts to the host are generated only under DSP software control.

The interrupt is generated by writing a 1 to the INTREQ bit in the DSP reset source/status register (RSTSRC). This causes the $\overline{\text{PINTA}}$ pin to be asserted on the local PCI bus, if the INTAM bit in the host status register (HSR) is 0. The $\overline{\text{PINTA}}$ pin is negated by writing a 1 to the INTRST bit in RSTSRC. The interrupt must be cleared by writing 1 to INTRST before another interrupt can be requested via INTREQ.

11 Boot Configuration for PCI Port

The following PCI port configurations, along with other device configurations, are determined at device reset via the boot configuration pins:

- ❏ EEPROM autoinitialization (EEAI) bits in EECTL: determines if PCI uses default values or read configure values from EEPROM.
- ❏ EEPROM size selection (EESZ) bits in EECTL (C62x only): determines EEPROM size.

For details on device and PCI boot configurations, refer to the device datasheet. The EEPROM interface is discussed in section 12.

The PCI port supports booting from the PCI bus. The CPU is stalled while the remainder of the device awakes from reset. During this period, the PCI host can initialize the DSP memory as necessary through the PCI. Once the PCI host is finished with all necessary initialization, it writes a 1 to the DSPINT bit in the host-to-DSP control register (HDCR) to release the DSP core from its stalled state. The DSP then begins execution from address 0h.

The sequence of events for PCI booting is:

1) PCI BOOTMODE is selected via configuration pins at reset. Refer to the device datasheet for details.
2) The PCI interface autoinitializes the PCI configuration registers via EEPROM (if selected).
3) PCI host sets memory and I/O enable.
4) The PCI master writes the DSP page register (DSPP).
5) The PCI master transfers the data to DSP memory-mapped space, starting at address 0h.
6) The PCI master can also access data memory, peripheral registers, EMIF.
7) The PCI master writes the DSPINT bit in HDCR with a 1 to release the DSP from reset.
8) DSP begins executing code from the program memory-mapped at 0h.

12 EEPROM Interface

The DSP supports the 4-wire serial EEPROM interface. The inter-integrated circuit (I^2C) and SPI interfaces are not supported. The EEPROM interface consists of the pins listed in Table 2.

Table 2. EEPROM Serial Interface Pins

Pin	Input/Output	Description
XSP_CLK	O	Serial EEPROM Clock
XSP_CS	O	Serial EEPROM Chip Select
XSP_DI†	I	Serial EEPROM Data In
XSP_DO	O	Serial EEPROM Data Out

† The XSP_DI pin should be pulled down.

The serial EEPROM clock is derived from the DSP peripheral clock. Normally, it is divided down by 2048 to drive the XSP_CLK pin. The XSP_CLK pin is only active during EEPROM access to minimize power.

For C62x devices, the state of the boot configuration pins, EESZ, at power-on reset determines if a serial EEPROM is present, and if so, what size. Table 3 summarizes the EEPROM sizes supported by the C62x devices. The C64x devices only support a 4K-bits EEPROM, and EESZ does not exist.

The EEPROM interface accesses the EEPROM as a 16-bit device only. The ORG pin of the EEPROM must be connected to V_{CC}.

Table 3. TMS320C62x DSP EEPROM Sizes Supported

EESZ Bits	EEPROM Size Supported on C62x DSP(bits)
000	No EEPROM present
001	1K
010	2K
011	4K
100	16K
100−111	Reserved

12.1 PCI Autoinitialization from EEPROM

The DSP allows some of the PCI configuration registers to be loaded from an external serial EEPROM. The PCI port without DSP intervention performs the autoinitialization process.

The state of the boot configuration pins EEAI and EESZ (C62x device only) at device reset determine if autoinitialization is enabled. Table 4 shows how the EEAI pin selects autoinitialization at reset.

Table 4. EEPROM Autoinitialization (EEAI)

EEAI Bit	Autoinitialization
0	Use default values
1	Read values from EEPROM

Autoinitialization is enabled if:

1) Configuration pin EEAI = 1 (autoinitialization is enabled).

2) Configuration pins EESZ ≠ 00b (indicates an EEPROM is present).

3) PCI operation is selected.

If any of these conditions are not met, default values are used for the PCI configuration registers and the EEPROM is not accessed. When all of these conditions are met, the contents from the EEPROM are loaded into some of the PCI configuration registers by the PCI interface (section 15.1). The size of the serial EEPROM is required to determine the serial protocol.

12.2 EEPROM Memory Map

The DSP requires a specific format for the data stored in the serial EEPROM. The first 28 bytes of the EEPROM are reserved for autoinitialization of PCI configuration registers. The remaining locations are not used for autoinitialization and can be used for storing other data. The EEPROM is always accessed as a 16-bit device. Table 5 summarizes the EEPROM memory map for the first 28 bytes. See also section 15.1 for details.

Table 5. EEPROM Memory Map

Address	Contents (msb … lsb)
0h	Vendor ID
1h	Device ID
2h	Class Code [7−0]/Revision ID
3h	Class Code [23−8]
4h	Subsystem Vendor ID
5h	Subsystem ID
6h	Max_Latency/Min_Grant
7h	PC_D1/PC_D0 (power consumed D1, D0)
8h	PC_D3/PC_D2 (power consumed D3, D2)
9h	PD_D1/PD_D0 (power dissipated D1, D0)
Ah	PD_D3/PD_D2 (power dissipated D3, D2)
Bh	Data_scale (PD_D3….PC_D0)
Ch	0000 0000 PMC[14−9], PMC[5], PMC[3]
Dh	Checksum

12.3 EEPROM Checksum

The configuration data contained in the EEPROM is checked against a checksum. The checksum is a 16-bit cumulative exclusive-OR (XOR) of the configuration data words contained in the EEPROM starting with an initial value of AAAAh. You must ensure that the proper 16-bit checksum value is written to address 0Dh when programming the EEPROM.

Checksum = AAAAh XOR Data(00h) XOR Data(01h)... XOR Data(0Ch)

If the checksum fails, the CFGERR bit in PCIIS and in HSR are set, and optionally, an interrupt to the DSP is generated. The DSP may or may not catch the interrupt, depending on the state of the core at the time. If the PCI is booting the device, the core is held in reset and will miss the interrupt.

The EEREAD bit in HSR is set, if EEPROM autoinitialization is used at power-on reset.

If the serial EEPROM is not accessed for PCI configuration purposes (that is, EEAI = 0, EESZ = 000b at reset), then the checksum is not performed.

Failed checksums result in the PCI configuration registers being initialized with default data. Refer to the specific PCI configuration registers to determine their default values (see section 15.1).

After successful PCI configuration register initialization (auto or default), the CFGDONE bit in RSTSRC is updated to allow the DSP to respond to reads, rather than terminating the cycle with disconnect retry.

12.4 DSP EEPROM Interface

The EEPROM can also be used by the DSP through three memory-mapped registers, EEPROM address register (EEADD), EEPROM data register (EEDAT), and EEPROM control register (EECTL). The DSP EEPROM interface is available immediately after reset. The CFGDONE bit in RSTSRC indicates when the EEPROM has been read for PCI autoinitialization.

Serial EEPROM device operation is controlled by seven instructions. The instruction opcode consists of two bits. Valid opcodes are presented in Table 6.

Table 6. EEPROM Command Summary

Op Code	Instruction	Description
10	READ	Reads data at specified address
00 (Address = 11xxxx)	EWEN	Write enable
11	ERASE	Erase memory at address
01	WRITE	Write memory at address
00 (Address = 10xxxx)	ERAL	Erases all memory locations
00 (Address = 01xxxx)	WRAL	Writes all memory locations
00 (Address = 00xxxx)	EWDS	Disables programming instructions

The EEPROM protocol is as follows:

1) Wait for the CFGDONE bit in RSTSRC to be set. The READY bit in EECTL and the EERDY bit in PCIIS are set as well.

2) Write EEPROM address to EEADD (address register, the EESZ determines which bits are significant).

3) For EEPROM reads, skip this step. For EEPROM writes (instruction WRITE/WRAL), write data to EEDAT. This data is immediately transferred to an internal register. Therefore, a DSP read from EEDAT returns invalid data.

4) Write the two-bit op code to the EECNT bits in EECTL.

5) The EEPROM interface then clocks out the EEPROM serial sequence.

6) Poll for READY = 1 in EECTL, or wait for interrupt (EERDY = 1 in PCIIS).

7) For EEPROM writes, skip this step. For EEPROM reads (instruction READ), read data from EEDAT.

The EEPROM serial sequence is initiated on writes to the EECNT bits. If the EECNT bits are written before the current command is complete (READY = 1), the command is executed after the current command completes. However, the EEDAT bits and the READY bit in EECTL from the previous command is corrupted. You should always poll for READY to be asserted before issuing new commands to the EEPROM controller.

13 Error Handling

The PCI configuration registers allow the DSP to handle error conditions. The following sections describe the handling of different error conditions.

13.1 PCI Parity Error Handling

If the DSP is mastering the bus, the data parity reported bit (bit 15) in the PCI status register (one of the PCI configuration registers) is set under one of the following conditions:

- Parity error during the data phase of a read transaction.
- $\overline{\text{PPERR}}$ has been asserted by the target during the data phase of a write transaction.

The data parity detected bit (bit 8) in the PCI status register is set under any of the following conditions:

- DSP is the PCI bus master and it detects a data parity error during a read transaction.
- DSP is the PCI bus target and it detects a data parity error during a write transaction.
- An address parity error is detected.

The PCI port asserts $\overline{\text{PPERR}}$ if the parity error reporting enable bit (bit 6) in the PCI command register (one of the PCI configuration registers) is set and the data parity detected bit (bit 8 in the PCI status register) is set. The assertion of $\overline{\text{PPERR}}$ remains valid until the second clock after the cycle in which the error occurred.

If a parity error is detected during a transfer involving DSP, the transaction is allowed to complete unless the PCI port is the master and a target disconnect is detected. The DSP will not master abort due to a parity error.

The PCI bus interface provides parity generation and verification for PCI bus data and PCI address parity. The PCI port asserts $\overline{\text{PSERR}}$ or $\overline{\text{PPERR}}$ for one PCI clock period and sets a flag in the PCI command register if it identifies a parity error.

13.2 PCI System Error Handling

An internal system error occurs if any of the following conditions are true:

- An address parity error is detected on the PCI bus (even if the DSP is not the target of the transaction) and the parity error reporting enable bit (bit 6) is set in the PCI command register (one of the PCI configuration registers).
- DSP detected that $\overline{\text{PPERR}}$ is asserted while mastering the bus.
- DSP received a target abort (disconnect without retry) while mastering the bus.

The DSP asserts $\overline{\text{PSERR}}$ if the system error reporting enable bit (bit 8) in the PCI command register is set, and the internal system error flag is set.

The DSP will halt and wait for software or hardware reset after $\overline{\text{PSERR}}$ has been asserted.

The DSP sets the signaled system error bit (bit 14) in the PCI status register whenever $\overline{\text{PSERR}}$ is asserted.

13.3 PCI Master Abort Protocol

In the event that a master abort occurs while the DSP is the master on the PCI bus, the current transfer is terminated on both the PCI bus and the auxiliary DMA or EDMA interface. The received master abort signal is set in the PCI status register. The PCIMASTER bit in PCIIS is set, and optionally an interrupt generated.

A received master abort resets the START bits in PCIMC to 00b. Any master transactions in progress stops.

13.4 PCI Target Abort Protocol

In the event that a target abort occurs while the DSP is the master on the PCI bus, the current transfer is terminated on the PCI bus and the auxiliary DMA or EDMA interface. The received target abort signal is set in the PCI status register. The PCITARGET bit in PCIIS is set, and optionally an interrupt is generated.

The target abort follows the same procedure as disabling a master transaction. The received target abort signal is used to reset the START bits in PCIMC to 00b. Further writes to the DSP memory is prevented. The interrupt indicates that the transfer did not succeed.

14 Power Management (C62x DSP only)

14.1 PCI Power Management

PCI Power Management Specification revision 1.1 defines power management states $D0_{unitialized}$, $D0_{active}$, D1, D2, $D3_{hot}$, and $D3_{cold}$. These power management states are:

$D0_{unitialized}$: Entered upon power up of the chip or any assertion of $\overline{PRST}$. The PCI configuration registers have not been initialized from EEPROM. When the DSP is in this state and autoinitialization is enabled, a PCI configuration register read or write generates a retry. If autoinitialization is not enabled, default values are loaded into the registers and PCI accesses can proceed normally. Once the configuration registers are loaded from EEPROM, the host can initialize the base register and I/O address register. This state is exited and enters $D0_{active}$ after the configuration registers have been initialized from EEPROM or default values, and the PCI I/O access enable bit (bit 0) and/or memory access enable bit (bit 1) in the PCI command register (one of the PCI configuration registers) are set.

$D0_{active}$: This is the normal operating state. In this state, the device supports full operation, and all peripherals are available. Transitions from $D0_{active}$ are accomplished by a power management request, $\overline{PRST}$ assertion, or removal of V_{DDcore}. If the transition from $D0_{active}$ is a power management request, the PCI can generate an interrupt to the DSP via the PWRMGMT bit in PCIIS.

D1: This is the first power management state. The exact operation of the chip in this mode is determined by DSP software. D1 power consumption is less than D0, but it is the function of the DSP software to reduce chip power. The memory and I/O access enable bits of the PCI configuration command register are disabled by a hardware mask to prevent memory and I/O cycles. The DSP responds to PCI configuration accesses. Transitions from D1 are accomplished by a power management request, $\overline{PRST}$ assertion, or removal of V_{DDcore}. When the transition from D1 is a power management request, an interrupt to the DSP can be generated via the PWRMGMT bit in PCIIS. This will allow wake up from DSP power down PD1 mode only, not PD2 or PD3.

D2: This is the second power management state. The exact operation of the chip in this mode is determined by DSP software. D2 power consumption is less than D1, but it is the function of the DSP software to reduce chip power. The memory and I/O access enable bits of the PCI configuration command register are disabled by a hardware mask to prevent memory and I/O cycles. The DSP responds to PCI configuration accesses. Transitions from D2 are accomplished by a power management request, $\overline{\text{PRST}}$ assertion, or removal of V_{DDcore}. When the transition from D2 is a power management request, an internal DSP warm reset is generated. This will allow wake up from DSP power down PD1, PD2, or PD3 mode.

$D3_{hot}$: This is the third power management state. The exact operation of the chip in this mode is determined by DSP software. $D3_{hot}$ power consumption should be less than that of D2, but it is the function of the DSP software to reduce chip power. The memory and I/O access enable bits of the PCI configuration command register are disabled by a hardware mask to prevent memory and I/O cycles. The DSP responds to PCI configuration accesses. Transitions from $D3_{hot}$ are accomplished by a power management request, $\overline{\text{PRST}}$ assertion, or removal of V_{DDcore}. When the transition from $D3_{hot}$ is a power management request, an internal DSP warm reset is generated. This will allow wake up from DSP power down PD1, PD2, or PD3 mode.

$D3_{cold}$: V_{DDcore} is removed and the device is totally shut down. The most significant power savings is achieved in this mode. A small amount of logic is powered from 3.3 V_{aux} to assert PME upon assertion of $\overline{\text{PWR\_WKP}}$. Transition from $D3_{cold}$ to $D0_{uninitialized}$ is accomplished by restoring V_{DDcore} and a $\overline{\text{RESET}}$ or $\overline{\text{PRST}}$.

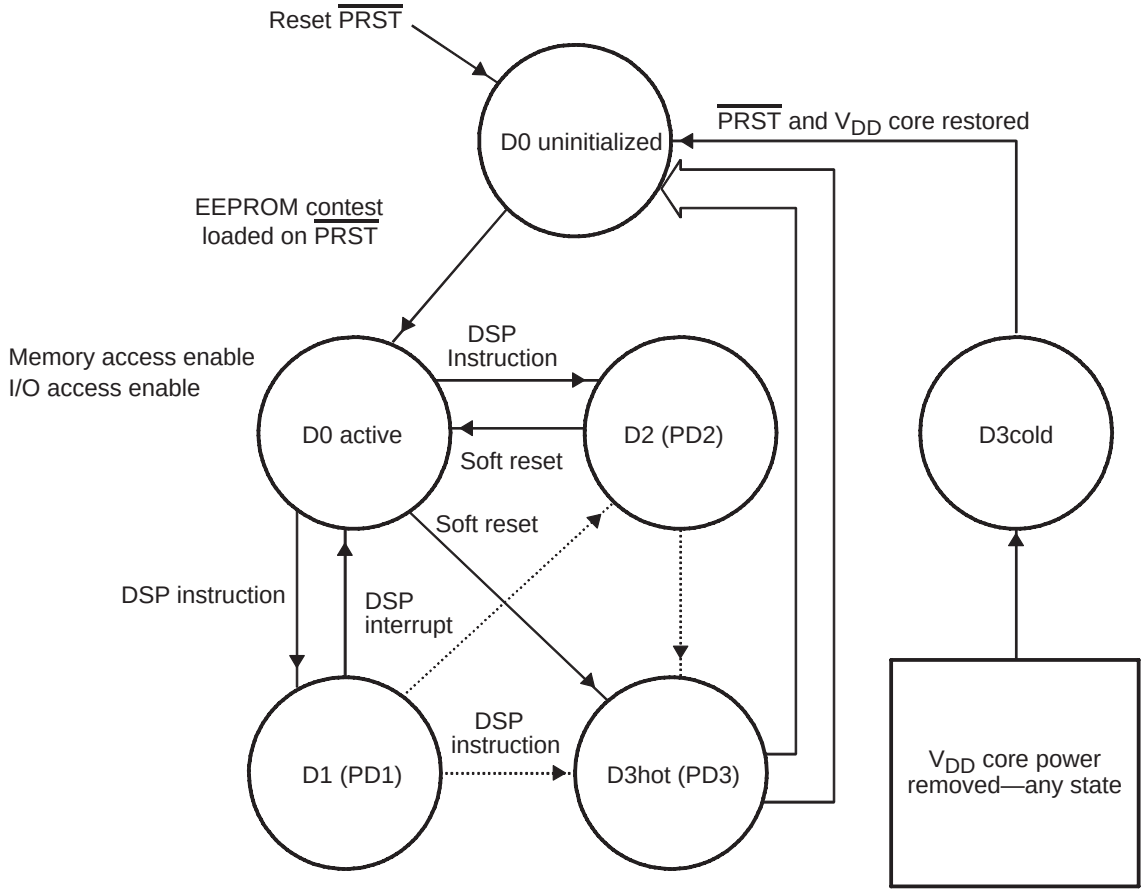
Figure 6 illustrates the DSP power state transitions. The DSP power management strategy is different from PCI Power Management Specification revision 1.1 in the following ways:

- DSP core power savings modes can only be changed by software instructions. To transition into a power-savings mode, the DSP core must first be fully operational in order to execute the software. Direct transitions from D1 to D2, from D1 to $D3_{hot}$, or from D2 to $D3_{hot}$ are not supported, as shown by the dotted lines in Figure 6. The core must always transition to $D0_{active}$ before proceeding to the desired power state.

- It is important to remember that the DSP software determines the power reduction mechanism for states D1, D2, and $D3_{hot}$.

Transitions from $D3_{hot}$ to D0 generate an internal DSP reset and an internal PCI reset. The requested state transitions to $D0_{uninitialized}$.

Figure 6. PCI Port Power Management State Transition Diagram



Since power management of the DSP is performed by DSP software, no power management is supported unless the DSP is booted from valid external memory or until the software is downloaded into internal DSP RAM and the DSP is released from reset. The DSP software, either internal or external, must be able to handle the host-initiated power management requests.

The PCI external host can not issue a PME until the DSP is in the $D0_{active}$ state, as is required by the Power Management Specification revision 1.1.

14.2 DSP Power Management Strategy

The DSP responds to PCI bus activity to trigger the DSP software-controlled transitions between the power management states. The DSP software is responsible for deciding the exact operation of the device in the power savings mode. The power management states are:

- $D0_{active}$: active state, no power savings.
- D1: DSP PD1 mode (CPU is halted, except for the interrupt logic).
- D2: DSP PD2 mode (CPU off, peripherals off).
- $D3_{hot}$: DSP PD3 mode (CPU off, peripherals off, PLL disabled).
- $D3_{cold}$: similar to $D3_{hot}$, except V_{DDcore} is now removed. A small amount of logic powered by 3.3 V_{aux} can be left powered to generate a PME on assertion of $\overline{PWR\_WKP}$.

The host requests for power state transitions by writing to the PWRSTATE bit in PMCSR (one of the PCI configuration registers). If the PWRSTATE bit is different than that of the CURSTATE bit in PMDCSR (one of the PCI peripheral registers), one of the following events occurs, depending on the current power management state of the DSP:

- **D0 or D1 state:** an interrupt is generated to wake up the core via the PWRMGMT bit in PCIIS. Upon wake up, the DSP software must read the requested state (REQSTATE bit in PMDCSR). The software must then update the CURSTATE bit in PMDCSR. At this time, the software can shut down peripherals, etc., as appropriate, with the execution of the appropriate power savings instruction. Direct transitions from D1 to D2, or from D1 to D3, are not supported because the DSP power savings are under software control.
- **D2 or D3 state:** a warm reset is generated to the DSP core. The warm reset wakes up the core from any power-savings mode that it may be in. The PCI I/O and memory access enable bits in the PCI command register maintain their states during the warm reset. After the core has waked up from the warm reset, it is in the $D0_{active}$ state. Upon wake up, the DSP software must read the request state (REQSTATE bit in PMDCSR). The software must then update the CURSTATE bit in PMDCSR. At this time, the software can shut down peripherals, etc., as appropriate, with the execution of the appropriate power savings instruction. Direct transitions from D2 to D3 are not supported because the DSP power savings are under software control.

The I/O access enable (bit 0) and memory access enable (bit 1) bits in the PCI command register are disabled by hardware in D1, D2, and D3 to disable memory and I/O cycles, as indicated in the Power Management Specification revision 1.1. If a PME to D0 is requested, the bits are enabled and are set to their original values.

14.3 DSP Resets

This section discusses the various types of resets, which are vital to the understanding of the DSP power management strategy.

Power on reset ($\overline{\text{RESET}}$): This is the pin reset applied during power up or hard reset. The state of the autoinitialization pins are sampled on the rising edge of the reset (EESZ[2−0], EEAI). All logic on the DSP is reset including the core and the peripherals. The EEPROM is autoinitialized on $\overline{\text{PRST}}$, not $\overline{\text{RESET}}$.

Warm reset: This reset applies to the core and peripherals. The PCI host can generate the warm reset by setting WARMRESET = 1 in the host-to-DSP control register (HDCR). The warm reset can also be generated on power management requests from D2 or D3. All logic on the DSP is reset including the core and the peripherals. WARMRESET is sufficient to wake the DSP from PD2 or PD3 power-down modes. However, the PCI configuration registers maintain their state. In addition, the memory access enable (bit 1) and I/O access enable (bit 0) bits in the PCI command register (one of the PCI configuration registers) are not affected by warm reset.

In summary, warm reset is generated by any one of the following:

- D2 or D3 power management requests (D2WARMONWKP and D3WARMONWKP bits set in PMDCSR)
- Write to the WARMRESET bit in HDCR by the PCI I/O during D0
- $\overline{\text{PWR\_WKP}}$ if D2 or D3 and D2WARMONWKP and D3WARMONWKP bits are set.

PCI reset ($\overline{\text{PRST}}$): This reset applies to the PCI bus interface unit (Figure 3, page 19) and the PCI FIFOs. PCI reset ($\overline{\text{PRST}}$) initializes the PCI configuration registers to their default states (if EEAI = 0) or autoinitializes them with values from the EEPROM (if EEAI = 1). The PCI base address registers (part of the PCI configuration registers), the memory access enable (bit 1), and the I/O access enable (bit 0) bits in the PCI command register are also reset by $\overline{\text{PRST}}$.

$\overline{\text{PRST}}$ is generated from the $\overline{\text{PRST}}$ pin, or from power management requests of D3 to D0.

The reset to the DSP core is the logical AND of the power-on reset ($\overline{\text{RESET}}$) and the warm reset, both of which are active low.

14.4 DSP Support for Power Management

The power management DSP control/status register (PMDCSR) is one of the PCI memory-mapped peripheral registers that allows power management control. The PMDCSR is discussed in section 15.3.2.

14.4.1 Power Management Control/Status Register (PMCSR) Bits

The power management control/status register (PMCSR) is discussed in section 15.1.23. The PMESTAT and PMEEN bits in PMCSR are powered from the 3.3 V_{aux} input. These bits are cleared on power-on transitions of 3.3 V_{aux}. $\overline{PRST}$, warm reset, and $\overline{RESET}$ do not affect the state of these bits; therefore, they are referred to as sticky bits.

In power managed PCs, the 3.3 V_{aux} is applied during $D3_{cold}$. Sticky bits are only reset during initial power up. In power nonmanaged PCs, the 3.3 V_{aux} transitions with device I/O power V_{DD}. Sticky bits are reset every time the system transitions from $D3_{cold}$ to D0. They are also reset on $\overline{PRST}$.

The PMESTAT bit is set to a 1 by assertion of $\overline{PWR\_WKP}$ when in $D3_{cold}$ ($\overline{RESET}$ active) or if not in $D3_{cold}$ by a DSP write to the PMESTAT bit in the power management DSP control/status register (PMDCSR), regardless of the value of the PMEEN bit.

The PMESTAT bit is cleared by writing a 1 to the PMESTAT bit in PMCSR. It is also cleared by a DSP write of 1 to the PMEEN bit in PMDCSR. PMCSR writes when PMESTAT is 0 have no effect on this bit. If the 3.3 V_{aux}DET pin is low (no support of PME assertion from $D3_{cold}$), then a $\overline{PRST}$ clears this bit. If the 3.3 V_{aux}DET pin is high (support of PME assertion from $D3_{cold}$), then a $\overline{RESET}$, $\overline{PRST}$, or warm reset does not affect this bit.

The PMEEN bit is set to 1 only when the PCI configuration register (PMCSR) is written with the PMEEN set to 1. The DSP cannot set the PMEEN bit via PMDCSR.

The PMEEN bit is cleared by a DSP write of 1 to the PMEEN bit in PMDCSR. PMCSR writes with PMEEN is 0 also clears the PMEEN bit. The DSP monitors the value of the PMEEN bit by reading the PMEEN bit in PMDCSR. If the 3.3 V_{aux}DET pin is low (no support of PME assertion from $D3_{cold}$), then a $\overline{PRST}$ clears this bit. If the 3.3 V_{aux}DET pin is high (support of PME assertion from $D3_{cold}$), then a $\overline{RESET}$, $\overline{PRST}$, or warm reset does not affect this bit.

The output of the PMEEN bit is used to prevent assertion of the PME pin when PMEEN is 0. The PME pin may be asserted only if PMEEN is set to 1.

The PMEDRVN bit in PMDCSR indicates to the DSP that the PME pin was driven active. This bit is cleared by DSP reads of PMDCSR only, but would be immediately set again if the PMEEN and PMESTAT bits in PMCSR are still both set.

14.4.2 3.3 V_{aux} Presence Detect Status Bit (AUXDETECT)

The 3.3 V_{aux}DET pin is used to indicate the presence of 3.3 V_{aux} when V_{DDcore} is removed. The DSP can monitor this pin by reading the AUXDETECT bit in PMDCSR. The PMEEN bit in PMCSR is held clear by the 3.3 V_{aux}DET pin being low.

14.4.3 PCI Port Response to $\overline{\text{PWR\_WKP}}$ and PME Generation

The PCI port responds differently to an active $\overline{\text{PWR\_WKP}}$ input, depending on whether V_{DDcore} is alive when 3.3 V_{aux} is alive. The PCI port response to $\overline{\text{PWR\_WKP}}$ is powered by 3.3 V_{aux}.

When V_{DDcore} is alive and 3.3 V_{aux} is alive (that is, all device power states but $D3_{cold}$), bits are set in the PCI interrupt source register (PCIIS) for the detection of the $\overline{\text{PWR\_WKP}}$ high-to-low and low-to-high transition. The $\overline{\text{PWR\_WKP}}$ signal is directly connected to the DSP PCI_WAKEUP interrupt. See section 10.

When V_{DDcore} is shut down and 3.3 V_{aux} is alive (in $D3_{cold}$), a $\overline{\text{PWR\_WKP}}$ transition causes the PMESTAT bit in PMCSR to be set (regardless of the PMEEN bit value). If the PMEEN bit is set, $\overline{\text{PWR\_WKP}}$ activity also causes the PME pin to be asserted and held active.

The PCI port can also generate PME depending on the HWPMECTL bits in PMDCSR. PME can be generated from any state or on transition to any state on an active $\overline{\text{PWR\_WKP}}$ signal, if the corresponding bit in the HWPMECTL bits is set.

Transitions on the $\overline{\text{PWR\_WKP}}$ pin can cause a CPU interrupt (PCI_WAKEUP, see section 10). The PWRHL and PWRLH bits in PCIIS indicate a high-to-low or low-to-high transition on the $\overline{\text{PWR\_WKP}}$ pin. If the corresponding interrupts are enabled in the PCI interrupt enable register (PCIIEN), a PCI_WAKEUP interrupt is generated to the CPU.

If 3.3 V_{aux} is not powered, the PME pin is in a high-impedance state. Once PME is driven active by the DSP, it is only deasserted when the PMESTAT bit in PMCSR is written with a 1 or the PMEEN bit is written with a 0. Neither $\overline{\text{PRST}}$, $\overline{\text{RESET}}$, or warm reset active can cause PME to go into a high-impedance state if it was already asserted before the reset.

14.4.4 DSP Interrupt Indicating that PWRSTATE has Changed

If a PMCSR write causes a change in the PWRSTATE bits, a CPU interrupt is generated. The PWRMGMT bit in PCIIS is set when the PWRSTATE bits are different than the current state. The CPU interrupt cannot be generated if the DSP clocks are not running. PMCSR changes due to $\overline{\text{RESET}}$ do not cause a CPU interrupt. A CPU interrupt occurs when $\overline{\text{PRST}}$ occurs, since a $\overline{\text{PRST}}$ assertion causes an implicit write of 0 to the PWRSTATE bits.

15 PCI Registers

There are three types of PCI registers:

- PCI Configuration Registers—accessible only by the external PCI host.
- PCI I/O Registers—accessible only by the external PCI host.
- PCI Memory-Mapped Registers—accessible by the DSP, and accessible by external PCI host via base address registers.

In addition, there are three resets related to the PCI registers. Each reset can have a different effect on the PCI registers, as indicated in the register descriptions in the following sections. The three resets are:

- **$\overline{\text{RESET}}$**—the main device reset pin
- **Warm Reset**—generated by the PCI host (or power management event)
- **$\overline{\text{PRST}}$**—the PCI reset signal

15.1 Configuration Registers

The DSP supports all standard PCI configuration registers. These registers, which can be accessed only by the external PCI host, contain the standard PCI configuration information (vendor identification, device identification, class code, revision number, base addresses, power management, etc.).

Depending on the boot and device configuration settings at device reset, the PCI configuration registers can be autoloaded from an EEPROM at power-on reset or can be initialized with default values at power-on reset.

If no EEPROM is present, the PCI configuration registers are initialized with their default values. If an EEPROM is present and autoinitialization is configured, the PCI configuration registers cannot be properly accessed by the host until they are fully read from the EEPROM. PCI host access to the PCI configuration registers before the completion of autoinitialization results in a disconnect with retry.

The CFGDONE and CFGERR bits in the PCI interrupt source register (PCIIS) and in the DSP reset source/status register (RSTSRC) indicate the status of the PCI configuration registers autoinitialization. See section 15.3.3 and section 15.3.1 for details. The PCI port asserts retries to prevent the host from performing any reads or writes to the PCI configuration registers until CFGDONE = 1, indicating that the PCI configuration registers have successfully been initialized with EEPROM or default values.

In addition, the EEPROM can also contain values for the power data, which can be selected by the power management control/status register (PMCSR). The power data stored in EEPROM is:

- PC_D0: Power consumed D0
- PC_D1: Power consumed D1
- PC_D2: Power consumed D2
- PC_D3: Power consumed D3
- PD_D0: Power dissipated D0
- PD_D1: Power dissipated D1
- PD_D2: Power dissipated D2
- PD_D3: Power dissipated D3

Table 7 lists the PCI configuration registers. Reads from the reserved fields return zeros, and writes to the reserved fields have no effect. These registers conform to the PCI Specification revision 2.2. Refer to this document and Power Management Specification revision 1.1 for more details on the registers and their operation.

The shaded registers in Table 7 can be autoloaded from the EEPROM at device power up. All registers are reset or loaded at PCI reset ($\overline{\text{PRST}}$).

Sections 15.1.1 to 15.1.24 summarize the bit fields in the PCI configuration registers. For more information, refer to PCI Specification revision 2.2.

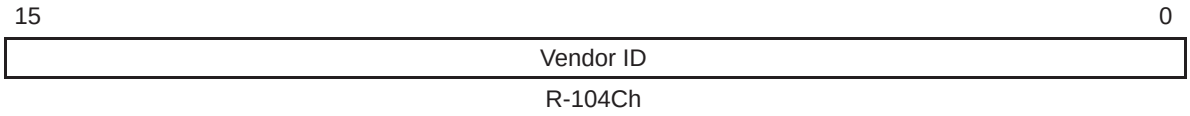
Table 7. PCI Configuration Registers

Address	Access	Byte 3	Byte 2	Byte 1	Byte 0
00h	read only	Device ID		Vendor ID	
04h	read/write	Status		Command	
08h	read only	Class Code			Revision ID
0Ch	read/write	Reserved	Header Type	Latency Timer	Cache Line Size
10h	read/write	Base 0 Address (4M-byte prefetchable)			
14h	read/write	Base 1 Address (8M-byte nonprefetchable)			
18h	read/write	Base 2 Address (4 words I/O)			
24h	read only	Reserved			
2Ch	read only	Subsystem ID		Subsystem Vendor ID	
30h	read only	Reserved			
34h	read only	Reserved			Capabilities Pointer
38h	read only	Reserved			
3Ch	read/write	Max_Latency	Min_Grant	Interrupt Pin	Interrupt Line
40h	read only	Power Management Capabilities		Next Item Pointer	Capability ID
44h	read/write	Power Data	Reserved	Power Management Control/Status	
48h FFh	read only	Reserved			

Note: Shaded registers can be autoloaded from EEPROM at autoinitialization.

15.1.1 Vendor Identification Register

Figure 7. Vendor Identification Register



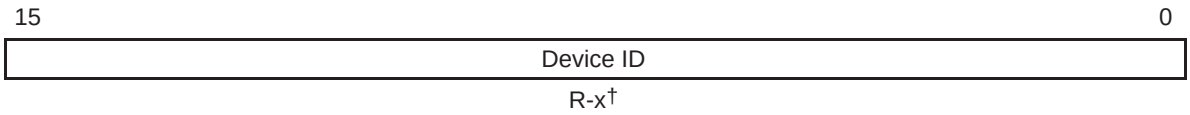
Legend: R = Read only; -*n* = value after reset

Table 8. Vendor Identification Register Field Descriptions

Bit	Field	Value	Description
15−0	Vendor ID	0−FFFFh	Vendor-defined device ID

15.1.2 Device Identification Register

Figure 8. Device Identification Register



Legend: R = Read only; -*n* = value after reset
† See the device-specific datasheet for the default value of this field.

Table 9. Device Identification Register Field Descriptions

Bit	Field	Value	Description
15−0	Device ID	0−FFFFh	Device manufacturer ID. See the device-specific datasheet for the value.

15.1.3 PCI Command Register

When the PWRSTATE bits in the power management control/status register (PMCSR) indicate D1, D2, or D3, the DSP must not respond to PCI activity to I/O or memory spaces and must not assert $\overline{\text{PINTA}}$. The DSP hardware monitors the PWRSTATE bits and masks the I/O access enable bit and the memory access enable bit in the PCI command register, and prevents $\overline{\text{PINTA}}$ assertion when the PWRSTATE bits indicate D1, D2, or D3. The PCI command register is shown in Figure 9 and described in Table 10.

Figure 9. PCI Command Register

15	10	9	8
Reserved†		Master back-to-back transaction enable	System error reporting enable
R-0		R/W-0	R/W-0

7	6	5	4	3	2	1	0
Data stepping control	Parity error reporting enable	VGA Palette Snoop	Memory Write and Invalidate enable	Special cycle recognition enable	Bus master functionality enable	Memory enable	I/O enable
R-0	R/W-0	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0

Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 10. PCI Command Register Field Descriptions

Bit	Value	Description
15−10	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
9	0−1	Master back-to-back transaction enable bit.
8	0−1	System error reporting enable bit.
7	0	Data stepping is not used.
6	0−1	Parity error reporting enable bit.
5	0	N/A: Not a VGA device.
4	0	Memory Write and Invalidate is not supported.
3	0	No special cycle recognition.
2	0−1	Master functionality bit.
1	0−1	Memory access enable bit.
0	0−1	I/O access enable bit.

15.1.4 PCI Status Register

Figure 10. PCI Status Register

15	14	13	12	11	10 9	8
Data parity reported error	Signaled system error	Received master abort	Received target abort	Signaled target abort	Device select signal timing	Master data parity detected error
R/WC-0	R/WC-0	R/WC-0	R/WC-0	R/WC-0	R-1	R/WC-0

7	6	5	4	3 0
Fast back-to-back capable	Reserved†	33 MHz maximum frequency	Capabilities list	Reserved†
R-0	R-0	R-0	R-1	R-0

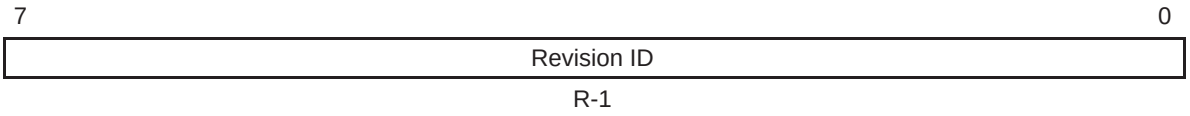
Legend: R = Read only; R/W = Read/Write; WC = Write 1 to reset, write of 0 has no effect; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 11. PCI Status Register Field Descriptions

Bit	Value	Description
15	0−1	Data parity reported error bit. Write 1 to reset the bit, a write of 0 has no effect.
14	0−1	Signaled system error bit. Write 1 to reset the bit, a write of 0 has no effect.
13	0−1	Received master abort bit. Write 1 to reset the bit, a write of 0 has no effect.
12	0−1	Received target abort bit. Write 1 to reset the bit, a write of 0 has no effect.
11	0−1	Signaled target abort bit. Write 1 to reset the bit, a write of 0 has no effect.
10−9	1	Device select signal timing: medium.
8	0−1	Master data parity detected error bit. Write 1 to reset the bit, a write of 0 has no effect.
7	0	Indicated the bit is not fast back-to-back capable.
6	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
5	0	33-MHz maximum frequency.
4	1	Capabilities list implemented (power management)
3−0	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.

15.1.5 Revision Identification Register

Figure 11. Revision Identification Register



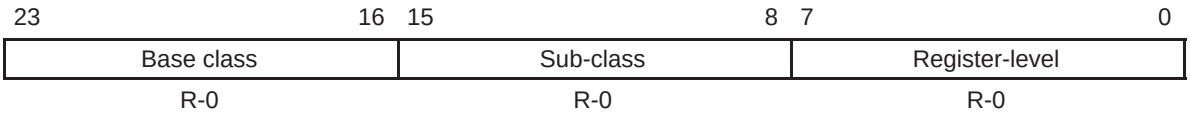
Legend: R = Read only; -*n* = value after reset

Table 12. Revision Identification Register Field Descriptions

Bit	Field	Value	Description
7−0	Revision ID	0−FFh	Device-specific revision ID.

15.1.6 Class Code Register

Figure 12. Class Code Register



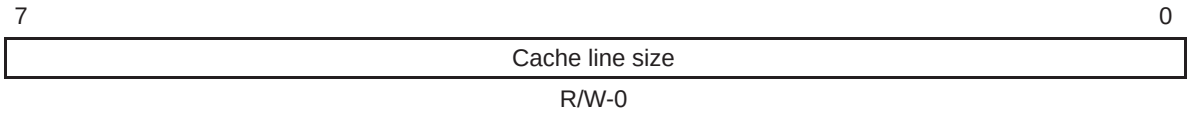
Legend: R = Read only; -*n* = value after reset

Table 13. Class Code Register Field Descriptions

Bit	Field	Value	Description
23−16	Base class	0h	Base class of device
15−8	Sub-class	0h	Sub-class
7−0	Register-level	0h	Register-level programming interface

15.1.7 Cache Line Size Register

Figure 13. Cache Line Size Register



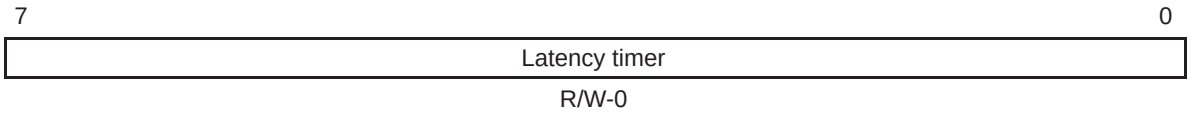
Legend: R/W = Read/Write; -*n* = value after reset

Table 14. Cache Line Size Register Field Descriptions

Bit	Field	Value	Description
7−0	Cache line size	0, 1h, 2h, 4h, 8h, 10h, 20h, 40h, 80h	Cache line size. This field only accepts power-of-2 cache line sizes. If a cache line size other than a power of 2 is written, 0 is written to this field.

15.1.8 Latency Timer Register

Figure 14. Latency Timer Register



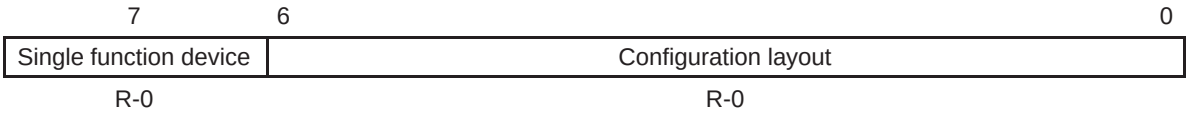
Legend: R/W = Read/Write; -*n* = value after reset

Table 15. Latency Timer Register Field Descriptions

Bit	Field	Value	Description
7−0	Latency timer	0−FFh	Latency timer.

15.1.9 Header Type Register

Figure 15. Header Type Register



Legend: R = Read only; -*n* = value after reset

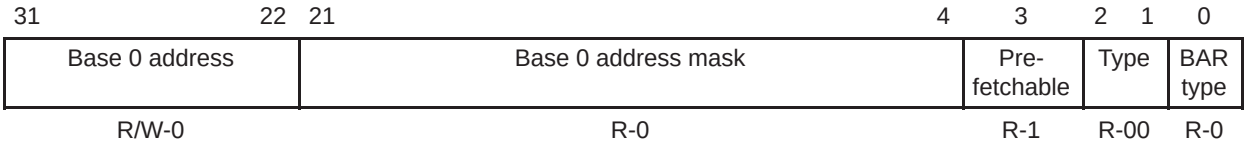
Table 16. Header Type Register Field Descriptions

Bit	Value	Description
7	0	Single function device.
6−0	0−7Fh	Configuration layout for typical PCI master/target device.

15.1.10 Base 0 Address Register

Mask for 4M bytes, prefetchable memory. The host reads FFC0 0008h after writing FFFF FFFFh. FFC0 0008h is used as the mask bits to determine the size of the base address region during register initialization.

Figure 16. Base 0 Address Register



Legend: R/W = Read/Write; -*n* = value after reset

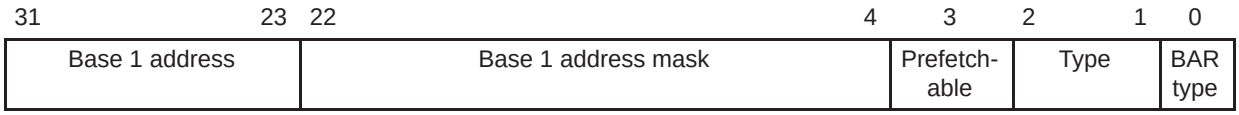
Table 17. Base 0 Address Register Field Descriptions

Bit	Value	Description
31−22	0−CFFh	Base 0 address. Writable bits. The host assigns the 4M PCI address space.
21−4	0	4M mask. These bits are read-only to indicate that this BAR requires 4M of address space.
3	1	Prefetchable. Indicates the BAR accesses prefetchable memory.
2−1	00	Type. This BAR must be located in the first 4G of address space.
0	0	BAR Type. This BAR is a memory BAR.

15.1.11 Base 1 Address Register

Mask for 8M bytes, nonprefetchable memory. The host reads FF80 0000h after writing FFFF FFFFh. FF80 0000h is used as the mask bits to determine the size of the base address region during register initialization.

Figure 17. Base 1 Address Register



Legend: R/W = Read/Write; -*n* = value after reset

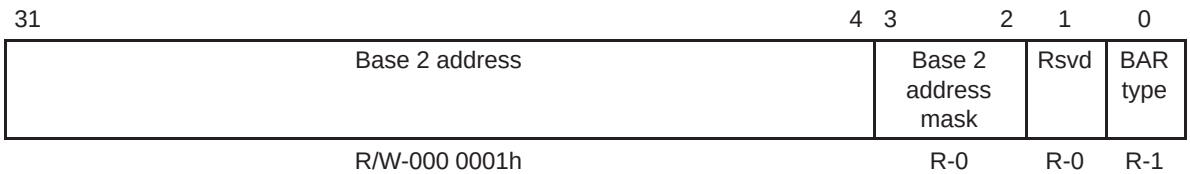
Table 18. Base 1 Address Register Field Descriptions

Bit	Value	Description
31−23	0−1FFh	Base 1 address. Writable bits. The host assigns the 8M PCI address space.
22−4	0	Base 1 address mask. These bits are read-only to indicate this BAR requires 8M of address space.
3	0	Prefetchable. Indicates that accesses through this BAR are not prefetchable.
2−1	0	Type. This BAR must be located in the first 4G of memory.
0	0	BAR type. This BAR is a memory BAR.

15.1.12 Base 2 Address Register

Mask for 16 bytes, I/O space. The host reads FFFF FFF1h after writing FFFF FFFFh. FFFF FFF1h is used as the mask bits to determine the size of the base address region during register initialization.

Figure 18. Base 2 Address Register



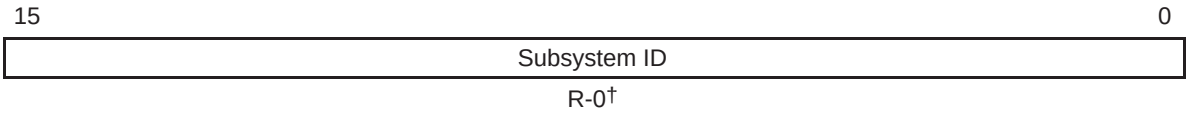
Legend: R/W = Read/Write; -*n* = value after reset

Table 19. Base 2 Address Register Field Descriptions

Bit	Value	Description
31−4	0−FFF FFFFh	Base 2 address. Writable bits. The host assigns the 16B IO address space.
3−2	0	Base 2 address mask. These bits are read-only to indicate that this BAR requires 16B of IO address space.
1	0	Reserved.
0	1	BAR type. This BAR is an IO BAR.

15.1.13 Subsystem Identification Register

Figure 19. Subsystem Identification Register



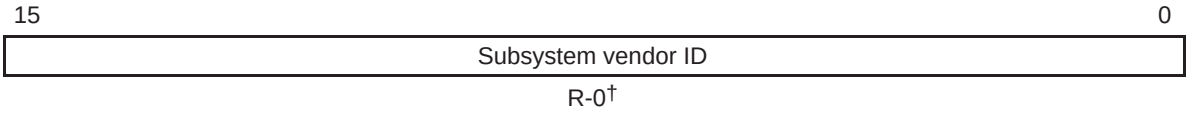
Legend: R = Read only; -*n* = value after reset
† Default can be set by EEPROM autoinitialization.

Table 20. Subsystem Identification Register Field Descriptions

Bit	Field	Value	Description
15−0	Subsystem ID	0−FFFFh	Add-in board or subsystem identifier.

15.1.14 Subsystem Vendor Identification Register

Figure 20. Subsystem Vendor Identification Register



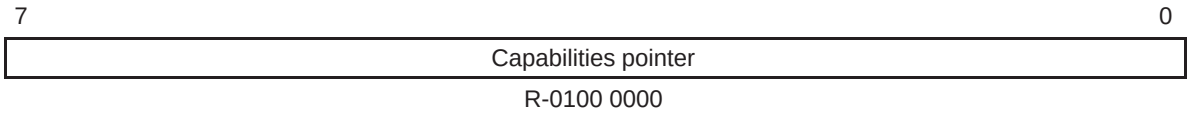
Legend: R = Read only; -*n* = value after reset
† Default can be set by EEPROM autoinitialization.

Table 21. Subsystem Vendor Identification Register Field Descriptions

Bit	Field	Value	Description
15−0	Subsystem vendor ID	0−FFFFh	Add-in board or subsystem vendor identifier.

15.1.15 Capabilities Pointer Register

Figure 21. Capabilities Pointer Register



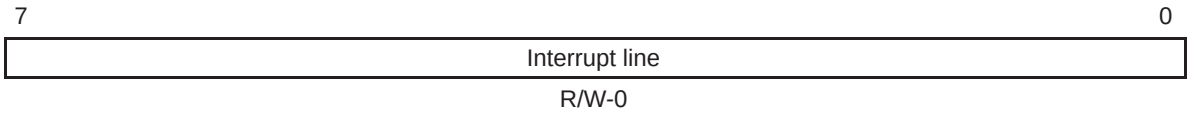
Legend: R = Read only; -*n* = value after reset

Table 22. Capabilities Pointer Register Field Descriptions

Bit	Field	Value	Description
7−0	Capabilities pointer	40h	Offset to the power management capability block.

15.1.16 Interrupt Line Register

Figure 22. Interrupt Line Register



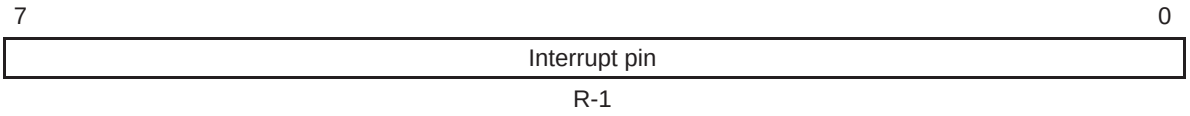
Legend: R/W = Read/Write; -*n* = value after reset

Table 23. Interrupt Line Register Field Descriptions

Bit	Field	Value	Description
7−0	Interrupt line	0−FFh	Interrupt line routing information.

15.1.17 Interrupt Pin Register

Figure 23. Interrupt Pin Register



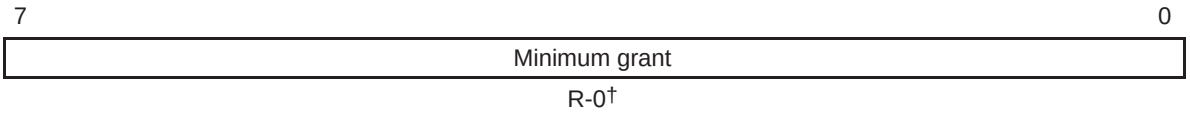
Legend: R = Read only; -*n* = value after reset

Table 24. Interrupt Pin Register Field Descriptions

Bit	Field	Value	Description
7−0	Interrupt pin	0−FFh	$\overline{\text{INTA}}$ is used for interrupts.

15.1.18 Min_Grant Register

Figure 24. Min_Grant Register



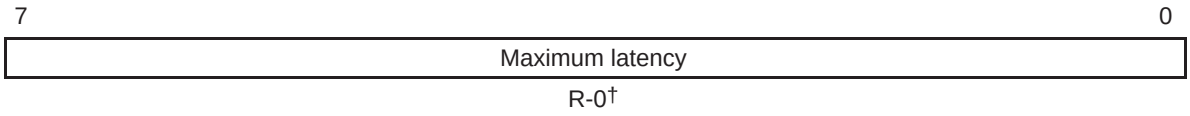
Legend: R = Read only; -*n* = value after reset
† Default can be set by EEPROM autoinitialization.

Table 25. Min_Grant Register Field Descriptions

Bit	Field	Value	Description
7−0	Minimum grant	0−FFh	Minimum grant bits.

15.1.19 Max_Latency Register

Figure 25. Max_Latency Register



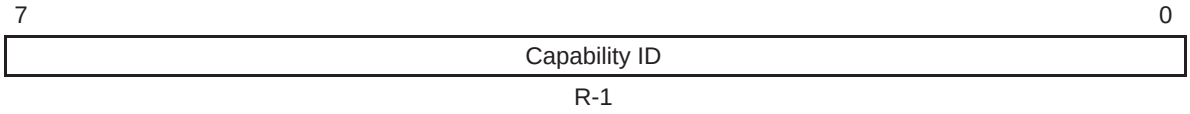
Legend: R = Read only; -*n* = value after reset
† Default can be set by EEPROM autoinitialization.

Table 26. Max_Latency Register Field Descriptions

Bit	Field	Value	Description
7−0	Maximum latency	0−FFh	Maximum latency bits.

15.1.20 Capability Identification Register

Figure 26. Capability Identification Register



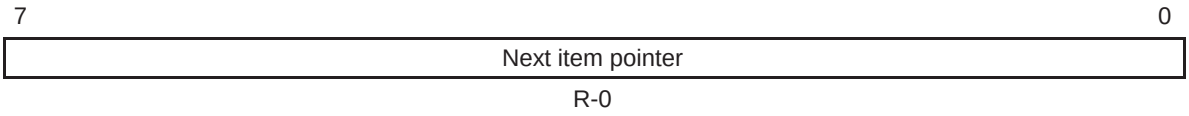
Legend: R = Read only; -*n* = value after reset

Table 27. Capability Identification Register Field Descriptions

Bit	Field	Value	Description
7−0	Capability ID	1	PCI power management identification bits.

15.1.21 Next Item Pointer Register

Figure 27. Next Item Pointer Register



Legend: R = Read only; -*n* = value after reset

Table 28. Next Item Pointer Register Field Descriptions

Bit	Field	Value	Description
7−0	Next item pointer	0	Next item in capabilities list pointer (0 is the last item).

15.1.22 Power Management Capabilities Register (PMC)

Figure 28. Power Management Capabilities Register (PMC)

15	14 11	10	9	8 6	5	4	3	2 0
PME_Support in D3	PME_Support	D2 Support	D1 Support	Auxiliary Supply Max Current	Initialization	Rsvd	PME Clock	Power Management
R-pin	R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-010

Legend: R = Read only; -*n* = value after reset; -pin = external pin value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 29. Power Management Capabilities Register (PMC) Field Descriptions

Bit	Field	Value	Description
15	PME_Support in D3		PME_Support in $D3_{cold}$ state bit. Read value depends on presence of 3.3 V on 3.3 V_{aux}DET pin.
		0	3.3 V_{aux}DET pin is low.
		1	3.3 V_{aux}DET pin is high.
14−11	PME_Support	0	PME_Support bits state that the board may assert PME.
10	D2 Support	0	D2_Support
9	D1 Support	0	D1_Support
8−6	Auxiliary Supply Max Current-	0	Auxiliary Supply Maximum Current bits. Because the PWRDATA register is implemented, this field returns 000.
5	Initialization	0	Device-Specific Initialization is or is not required.
4	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3	PME Clock	0	PME Clock bit. PCI clock is not required to assert PME.
2−0	Power Management	3h	Power Management Version bits.

15.1.23 Power Management Control/Status Register (PMCSR)

The power management control/status register (PMCSR) is shown in Figure 29 and described in Table 30.

Figure 29. Power Management Control/Status Register (PMCSR)

15	14 13	12 9	8	7 2	1 0
PMESTAT	DATASCALE	DATASEL	PMEEN	Reserved†	PWRSTATE
R/WC-0	R-0	R/W-0	R/W-0	R-0	R/W-0

Legend: R = Read only; R/W = Read/Write; WC = Write 1 to clear, write of 0 has no effect; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 30. Power Management Control/Status Register (PMCSR) Field Descriptions

Bit	Field	Value	Description
15	PMESTAT		Power management event status bit. If 3.3V_{aux}DET is low, this bit resets to 0; if 3.3V_{aux}DET is high, this bit is 0 on power reset.
			PMESTAT is a sticky bit (that is, the value is maintained when the main PCI bus power is off) powered from the PCI 3.3V_{aux} pin. Write 1 to clear the bit, a write of 0 has no effect.
		0	No power management event has occurred.
		1	Power management event has occurred. If PMEEN is 1, PME pin is also asserted.
14−13	DATASCALE	0−3h	Data scale bits. Scaling factor for data read from PWRDATA. This field changes with DATASEL values and is initialized with PWRDATA initialization data.
		0	Reserved
		1h	0.1 Watt × value from PWRDATA
		2h	0.01 Watt × value from PWRDATA
		3h	0.001 Watt × value from PWRDATA

Table 30. Power Management Control/Status Register (PMCSR) Field Descriptions (Continued)

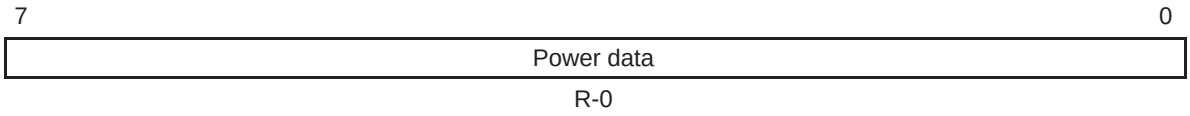
Bit	Field	Value	Description
12−9	DATASEL	0−Fh	Data select bits. Select bits for PWRDATA register and DATASCALE field.
		0	PWRDATA reads return D0 power consumed.
		1h	PWRDATA reads return D1 power consumed.
		2h	PWRDATA reads return D2 power consumed.
		3h	PWRDATA reads return $D3_{hot}$ power consumed.
		4h−7h	PWRDATA reads return 0.
		8h	PWRDATA reads return D0 power dissipated.
		9h	PWRDATA reads return D1 power dissipated.
		Ah	PWRDATA reads return D2 power dissipated.
		Bh	PWRDATA reads return $D3_{hot}$ power dissipated.
		Ch−Fh	PWRDATA reads return 0.
8	PMEEN		PME assertion enable bit. If $3.3V_{aux}$DET is low, this bit resets to 0; if $3.3V_{aux}$DET is high, this bit is 0 on power reset. PMEEN is a sticky bit (that is, the value is maintained when the main PCI bus power is off) powered from the PCI $3.3V_{aux}$ pin.
		0	PME assertion is disabled.
		1	PME assertion is enabled.
7−2	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.

Table 30. Power Management Control/Status Register (PMCSR) Field Descriptions (Continued)

Bit	Field	Value	Description
1−0	PWRSTATE	0−3h	Power state bits. Reads of PWRSTATE return the current power state as provided by the CURSTATE bit in the power management DSP control/status register (PMDCSR). Writes of PWRSTATE are the requested power state by the host. Writes when the bits are not the same as the value currently in the REQSTATE bits of PMDCSR cause a power management interrupt or warm reset to the DSP and update the REQSTATE bits with the value from these bits. Writes when the value is the same as the current value of the REQSTATE bits do not cause an interrupt. Only power state writes to legal transitions are processed. All writes terminate properly on the PCI bus, but illegal transitions are not processed by the core. If the software attempts to write an unsupported, optional state to this field, the write operation must complete normally on the PCI bus; however, the data is discarded and no state change occurs.
		0	Power state D0
		1h	Power state D1
		2h	Power state D2
		3h	Power state D3

15.1.24 Power Data Register (PWRDATA)

Figure 30. Power Data Register (PWRDATA)



Legend: R = Read only; -*n* = value after reset

Table 31. Power Data Register (PWRDATA) Field Descriptions

Bit	Field	Value	Description
7−0	Power data	0−Fh	Power data bits. Reads return the power consumed or dissipated in the device power management state as selected by the DATASEL bits in PMSCR.
		0	D0 power consumed.
		1h	D1 power consumed.
		2h	D2 power consumed.
		3h	$D3_{hot}$ power consumed.
		4h	D0 power dissipated.
		5h	D1 power dissipated.
		6h	D2 power dissipated.
		7h	$D3_{hot}$ power dissipated.
		8h−Fh	Reads return 0.

15.2 I/O Registers

Table 32 lists the PCI I/O registers located in the PCI host I/O space. They can be accessed only by the PCI host in the base 1 or base 2 address ranges specified in the base 1 address register and the base 2 address register, respectively. The base 1 address register and base 2 address register are PCI configuration registers, described in section 15.1. The locations of the different base addresses are discussed in section 3. All PCI I/O registers are byte-addressable. Table 33 shows the PCI I/O register locations accessed via the I/O base address (base 2 address).

For processors that do not support I/O access, it can also access the PCI I/O registers via nonprefetchable reads and writes to the base 1 memory. Refer to the device-specific datasheet for the correct address of these registers when accessed via base 1. The DSP cannot access the I/O registers at these locations. They are accessible only by the PCI host. No memory reads/writes or DSP memory-mapped register access occurs at these locations for base 1 access. Base 0 access to these locations is mapped to the DSP memory-mapped registers (not I/O registers).

Table 32. PCI I/O Registers

Acronym	Register Name	Section
HSR	Host status register	15.2.1
HDCR	Host-to-DSP control register	15.2.2
DSPP	DSP page register	15.2.3

Table 33. PCI I/O Registers Accessed via I/O Space (Base 2 Memory)

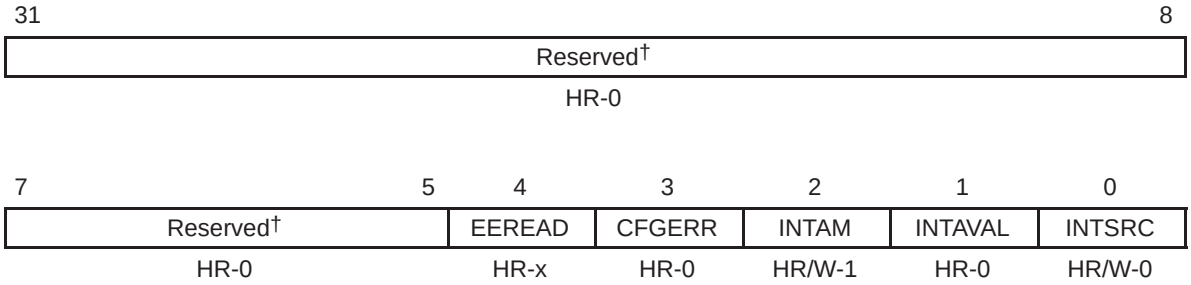
Address†	Register/Port Accessed
I/O Base Address + 00h	Host status register (HSR)
I/O Base Address + 04h	Host-to-DSP control register (HDCR)
I/O Base Address + 08h	DSP page register (DSPP)
I/O Base Address + 0Ch	Reserved

† I/O Base Address is specified in the Base 2 Address Register. See Table 7, page 52.

15.2.1 Host Status Register (HSR)

The host status register (HSR) is shown in Figure 31 and described in Table 34.

Figure 31. Host Status Register (HSR)



Legend: H = Host access; R = Read only; R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset
† If writing to this field, always write the default value for future device compatibility.

Table 34. Host Status Register (HSR) Field Descriptions

Bit	Field	Value	Description
31−5	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
4	EEREAD		Indicates if the PCI configuration registers were initialized from EEPROM. Host read-only bit, writes have no effect. Reset source is $\overline{\text{PRST}}$.
		0	PCI configuration registers were initialized from default values.
		1	PCI configuration registers were initialized from EEPROM.
3	CFGERR		Indicates if an autoinitialization configuration error occurred. Host read-only bit, writes have no effect. Reset source is $\overline{\text{PRST}}$.
		0	No error occurred.
		1	Autoinitialization configuration error occurred.
2	INTAM		$\overline{\text{PINTA}}$ mask. Disables DSP assertion of $\overline{\text{PINTA}}$. Only written by the PCI host (during D1, D2, or D3, the $\overline{\text{PINTA}}$ is masked by the power management logic). Reset source is $\overline{\text{PRST}}$.
		0	$\overline{\text{PINTA}}$ is asserted by the DSP when the INTREQ bit in the DSP reset source/status register (RSTSRC) is set.
		1	$\overline{\text{PINTA}}$ is not asserted. Note that this does not reset the interrupt generating logic, you must set the INTRST bit in the DSP reset source/status register (RSTSRC).

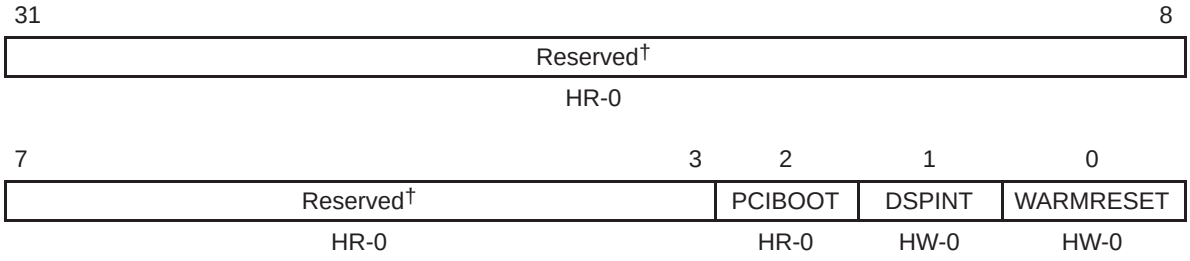
Table 34. Host Status Register (HSR) Field Descriptions (Continued)

Bit	Field	Value	Description
1	INTAVAL		Indicates the current PINTA pin value. Host read-only bit, writes have no effect. PINTA can be deasserted by the PCI host writing a 1 to the INTSRC bit or by the DSP writing a 1 to the INTRST bit in the DSP reset source/status register (RSTSRC). Reset source is PRST.
		0	PINTA is not asserted (inactive).
		1	PINTA is asserted (active).
0	INTSRC		PCI IRQ source active since last HSR clear. This bit, when 1, indicates that the DSP asserted the PINTA interrupt by writing the INTREQ bit in the DSP reset source/status register (RSTSRC), and the INTAM bit was 0. Reset source is PRST. This bit can be cleared by the PCI host by writing a 1 to this bit. This also negates the PINTA signal.
		0	For reads, PINTA was not asserted after last clear. For writes, no affect.
		1	For reads, PINTA was asserted after last clear. For writes, deassert PINTA. Note that this does not enable future interrupts. The INTRST bit in RSTSRC must also be set to allow future interrupts. See section 15.3.1.

15.2.2 Host-to-DSP Control Register (HDCR)

The host-to-DSP control register (HDCR) is shown in Figure 32 and described in Table 35.

Figure 32. Host-to-DSP Control Register (HDCR)



Legend: H = Host access; R = Read only; W = Write only; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

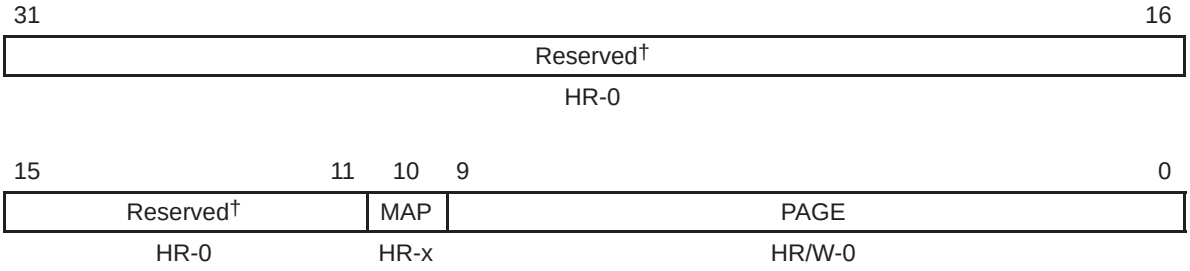
Table 35. Host-to-DSP Control Register (HDCR) Field Descriptions

Bit	Field	Value	Description
31−3	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
2	PCIBOOT		PCI boot mode. Host read-only bit, writes have no effect. Reset source is RESET.
		0	DSP does not boot from the PCI.
		1	DSP boots from the PCI.
1	DSPINT		DSP interrupt. Host write-only bit, reads return 0.
			The interrupt is generated to the core via the HOSTSW bit in the PCI interrupt source register (PCIIS). If booting from the PCI interface, this interrupt takes the core out of reset. In all other cases, the DSP core must have its clock running and the HOSTSW bit in the PCI interrupt enable register (PCIIEN) unmasked in order to latch the interrupt. Reset source is PRST.
		0	A write of 0 is ignored.
		1	Generates a host interrupt to the DSP.
0	WARMRESET		DSP warm reset. Host write-only bit.
			WARMRESET only applies in D0. WARMRESET should not be used if the core is in power management state D1, D2, or D3 (I/O access is disabled in these states). Reset source is RESET.
		0	A write of 0 is ignored.
		1	Resets the DSP. The DSP is held in reset for 16 PCI cycles. The DSP cannot be accessed until 16 PCI clocks after WARMRESET is written.

15.2.3 DSP Page Register (DSPP)

The DSP page register (DSPP) is shown in Figure 33 and described in Table 36.

Figure 33. DSP Page Register (DSPP)



Legend: H = Host access; R = Read only; R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset
† If writing to this field, always write the default value for future device compatibility.

Table 36. DSP Page Register (DSPP) Field Descriptions

Bit	Field	Value	Description
31−11	Reserved	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
10	MAP		Indicates the memory map being used by the DSP (C62x DSP only). Host read-only bit, writes have no effect. Reset source is RESET.
		0	Map 0
		1	Map 1
9−0	PAGE	0−3FFh	Locates a 4M-byte memory window within the DSP address map for prefetchable (base 0) memory accesses. Reset source is PRST.

15.3 Memory-Mapped Registers

Table 37 lists the memory-mapped registers for DSP control/status of the PCI interface. These registers can be accessed by both the DSP and the PCI host. See the device-specific datasheet for the memory address of these registers.

Table 37. PCI Memory-Mapped Registers

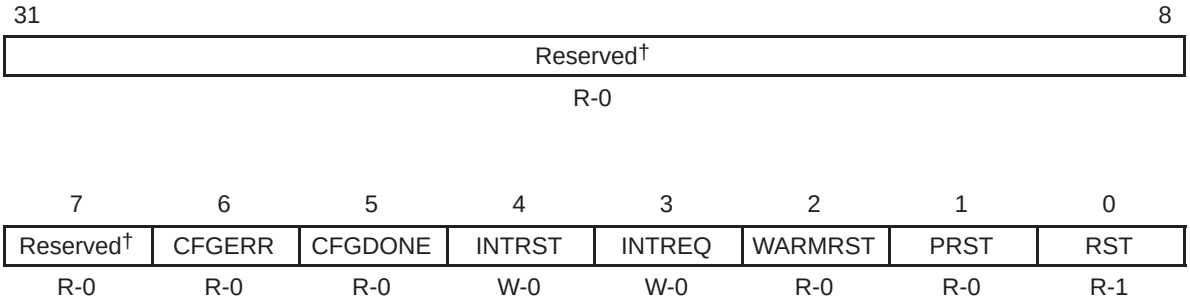
Acronym	Register Name	Section
RSTSRC	DSP reset source/status register	15.3.1
PMDCSR†	Power management DSP control/status register	15.3.2
PCIIS	PCI interrupt source register	15.3.3
PCIIEN	PCI interrupt enable register	15.3.4
DSPMA	DSP master address register	15.3.5
PCIMA	PCI master address register	15.3.6
PCIMC	PCI master control register	15.3.7
CDSPA	Current DSP address register	15.3.8
CPCIA	Current PCI address register	15.3.9
CCNT	Current byte count register	15.3.10
EEADD	EEPROM address register	15.3.11
EEDAT	EEPROM data register	15.3.12
EECTL	EEPROM control register	15.3.13
HALT†	PCI transfer halt register	15.3.14
TRCTL‡	PCI transfer request control register	15.3.15

† This register only applies to C62x DSP.
‡ TRCTL register only applies to C64x DSP.

15.3.1 DSP Reset Source/Status Register (RSTSRC)

The DSP reset source/status register (RSTSRC) shows the reset status of the DSP. It gives the DSP visibility to which reset source caused the last reset. The RSTSRC is shown in Figure 34 and described in Table 38. The RST, PRST, and WARMRST bits are cleared by a read of RSTSRC.

Figure 34. DSP Reset Source/Status Register (RSTSRC)



Legend: R = Read only; W = Write only; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 38. DSP Reset Source/Status Register (RSTSRC) Field Descriptions

Bits	***field***†	***symval***†	Value	Description
31−7	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
6	CFGERR	OF(*value*)		Configuration error bit. An error occurred when trying to load the configuration registers from EEPROM (Checksum failure). Read-only bit, writes have no effect. The reset source is PRST.
		DEFAULT	0	No configuration error.
			1	Checksum error during EEPROM autoinitialization.
5	CFGDONE	OF(*value*)		Configuration hold bit. EEPROM has finished loading the PCI configuration registers. Read-only bit, writes have no effect. The reset source is RESET.
		DEFAULT	0	Configuration registers have not been loaded.
			1	Configuration registers load from EEPROM is complete.

† For CSL implementation, use the notation PCI_RSTSRC_*field_symval*

Table 38. DSP Reset Source/Status Register (RSTSRC) Field Descriptions (Continued)

Bits	*field*†	*symval*†	Value	Description
4	INTRST	OF(*value*)		PINTA reset bit. This bit must be asserted before another host interrupt can be generated. Write-only bit, reads return 0. The reset sources are RESET and Warm.
		DEFAULT	0	Writes of 0 have no effect.
		NO		
		YES	1	When a 1 is written to this bit, PINTA is deasserted and the interrupt logic is reset to enable future interrupts.
3	INTREQ	OF(*value*)		Request a DSP-to-PCI interrupt when written with a 1. Write-only bit, reads return 0. The reset sources are RESET and Warm.
		DEFAULT	0	Writes of 0 have no effect.
		NO		
		YES	1	Causes assertion of PINTA if the INTAM bit in the host status register (HSR) is 0.
2	WARMRST	OF(*value*)		A host software reset of the DSP or a power management warm reset occurred since the last RSTSRC read or last RESET. Read-only bit, writes have no effect. This bit is set by a host write of 0 to the WARMRESET bit in the host-to-DSP control register (HDCR) or a power management request from D2 or D3. Cleared by a read of RSTSRC or RESET assertion. The reset source is RESET.
		DEFAULT	0	No warm reset since last RSTSRC read or RESET.
			1	Warm reset since last RSTSRC read or RESET.
1	PRST	OF(*value*)		Indicates occurrence of a PRST reset since the last RSTSRC read or RESET assertion. Read-only bit, writes have no effect. Cleared by a read of RSTSRC or RESET active. When PRST is held active (low), this bit always reads as 1. The reset source is RESET.
		DEFAULT	0	No PRST reset since last RSTSRC read.
			1	PRST reset has occurred since last RSTSRC read.

† For CSL implementation, use the notation PCI_RSTSRC_*field_symval*

Table 38. DSP Reset Source/Status Register (RSTSRC) Field Descriptions (Continued)

Bits	*field*†	*symval*†	Value	Description
0	RST	OF(*value*)		Indicates a device reset ($\overline{\text{RESET}}$) occurred since the last RSTSRC read. Read-only bit, writes have no effect.
				Cleared by a read of RSTSRC. The reset source is $\overline{\text{RESET}}$.
			0	No device reset ($\overline{\text{RESET}}$) since last RSTSRC read
		DEFAULT	1	Device reset ($\overline{\text{RESET}}$) has occurred since last RSTSRC read

† For CSL implementation, use the notation PCI_RSTSRC_*field_symval*

15.3.2 Power Management DSP Control/Status Register (PMDCSR) (C62x DSP only)

The power management DSP control/status register (PMDCSR) allows power management control. The PMDCSR is shown in Figure 35 and described in Table 39.

Figure 35. Power Management DSP Control/Status Register (PMDCSR)

31 19	18 11	10	9	8
Reserved†	HWPMECTL	D3WARMONWKP	D2WARMONWKP	PMEEN
R-0	R/W-1000 1000	R-x	R-x	R/W-x

7	6	5	4	3 2	1 0
PWRWKP	PMESTAT	PMEDRVN	AUXDETECT	CURSTATE	REQSTATE
R-x	R-x/W-0	R-0	R-x	R/W-0	R-0

Legend: R = Read only; R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset
† If writing to this field, always write the default value for future device compatibility.

Table 39. Power Management DSP Control/Status Register (PMDCSR) Field Descriptions

Bits	*field*†	*symval*†	Value	Description
31−19	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
18−11	HWPMECTL	OF(*value*)	0−FFh	Hardware PME control. Allows PME to be generated automatically by hardware on active $\overline{\text{PWR\_WKP}}$ if the corresponding bit is set. The reset source is $\overline{\text{RESET}}$.
		−	0	Reserved
		REQD0	1h	Requested state = 00
		REQD1	2h	Requested state = 01
		REQD2	3h	Requested state = 10
		REQD3	4h	Requested state = 11
		−	5h−87h	Reserved
		DEFAULT	88h	Reserved
		−	89h−FFh	Reserved
10	D3WARMONWKP	OF(*value*)		Warm reset from D3. Read-only bit, writes have no effect. Warm resets are only generated from $\overline{\text{PWR\_WKP}}$ if the following conditions are true: ❑ $\overline{\text{PRST}}$ (PCI reset) is deasserted ❑ PCLK is active. The reset source is $\overline{\text{RESET}}$.
		DEFAULT	0	No warm reset is generated on $\overline{\text{PWR\_WKP}}$ asserted (low).
			1	Warm reset is generated on $\overline{\text{PWR\_WKP}}$ asserted if the current state is D3.

† For CSL implementation, use the notation PCI_PMDCSR_*field_symval*

Table 39. Power Management DSP Control/Status Register (PMDCSR) Field Descriptions (Continued)

Bits	*field*†	*symval*†	Value	Description
9	D2WARMONWKP	OF(*value*)		Warm reset from D2. Read-only bit, writes have no effect. Warm resets are only generated from $\overline{\text{PWR\_WKP}}$ if the following conditions are true: ❑ $\overline{\text{PRST}}$ (PCI reset) is deasserted ❑ PCLK is active. The reset source is $\overline{\text{RESET}}$.
		DEFAULT	0	No warm reset is generated on $\overline{\text{PWR\_WKP}}$ asserted (low).
			1	Warm reset is generated on $\overline{\text{PWR\_WKP}}$ asserted if the current state is D2.
8	PMEEN	OF(*value*)		PME assertion enable bit. Reads return current value of PMEEN bit in the power management control/status register (PMCSR). Writes of 1 clear both the PMEEN and PMESTAT bits in PMCSR, writes of 0 have no effect.
		DEFAULT	0	PMEEN bit in PMCSR is 0; PME assertion is disabled.
		CLR	1	PMEEN bit in PMCSR is 1; PME assertion is enabled. The reset sources are $\overline{\text{RESET}}$ and Warm.
7	PWRWKP	OF(*value*)		$\overline{\text{PWRWKP}}$ pin value. Read-only bit, writes have no effect. There are no reset sources.
		DEFAULT	0	$\overline{\text{PWR\_WKP}}$ pin is low.
			1	$\overline{\text{PWR\_WKP}}$ pin is high.
6	PMESTAT	OF(*value*)		PMESTAT sticky bit value. Reads return the current status of the PMESTAT bit in the power management control/status register (PMCSR). If the PMESTAT and PMEEN bits are written with a 1 at the same time, the PMEEN and PMESTAT bits are cleared. Writes of 0 have no effect.
		DEFAULT	0	No effect.
		SET	1	Forces the PMESTAT bit in PMCSR to 1. The reset sources are $\overline{\text{RESET}}$ and Warm.

† For CSL implementation, use the notation PCI_PMDCSR_*field_symval*

Table 39. Power Management DSP Control/Status Register (PMDCSR) Field Descriptions (Continued)

Bits	*field*†	*symval*†	Value	Description
5	PMEDRVN	OF(*value*)		PME driven high. The DSP has driven the PME pin active high. Read-only bit, writes have no effect. The reset sources are RESET and Warm.
		DEFAULT	0	DSP read of PMDCSR, but bit would be set if the PMEEN and PMESTAT bits are both still high.
			1	PMEEN and PMESTAT bits in the power management control/status register (PMCSR) are high.
4	AUXDETECT	OF(*value*)		3.3V_{aux}DET pin value. Read-only bit, writes have no effect. The reset sources are RESET and Warm.
		DEFAULT	0	3.3 V_{aux}DET is low.
			1	3.3 V_{aux}DET is high.
3−2	CURSTATE	OF(*value*)	0−3h	Current power state. Reflects the current power management state of the device. On changing state, the device must change the CURSTATE bits. The value written here is used for PCI reads of the PWRSTATE bits in the power management control/status register (PMCSR). The reset source is RESET (unaffected by PRST or Warm reset).
		DEFAULT	0	Current state = 00
		D0		
		D1	1h	Current state = 01
		D2	2h	Current state = 10
		D3	3h	Current state = 11
1−0	REQSTATE	OF(*value*)	0−3h	Last requested power state. Last value written by the host to the PCI PWRSTATE bits in the power management control/status register (PMCSR). Cleared to 00b on RESET or PRST. Read-only bit, writes have no effect. The reset sources are RESET and PRST.
		DEFAULT	0	

† For CSL implementation, use the notation PCI_PMDCSR_*field_symval*

15.3.3 PCI Interrupt Source Register (PCIIS)

The PCI interrupt source register (PCIIS) shows the status of the interrupt sources. Writing a 1 to the bit(s) clears the condition. Writes of 0 to, and reads from, the bit(s) have no effect. The PCIIS is shown in Figure 36 and described in Table 40.

Figure 36. PCI Interrupt Source Register (PCIIS)

31	16
Reserved†	
R-0	

15	13	12	11	10	9	8
Reserved†		DMAHALTED‡	PRST	Reserved†	EERDY	CFGERR
R-0		R/W-0	R/W-0	R-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
CFGDONE	MASTEROK	PWRHL	PWRLH	HOSTSW	PCIMASTER	PCITARGET	PWRMGMT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.
‡ This bit is reserved on C64x DSP.

Table 40. PCI Interrupt Source Register (PCIIS) Field Descriptions

Bit	***field***†	***symval***†	Value	Description
31−13	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	DMAHALTED	OF(*value*)		DMA transfers halted enable bit (C62x DSP only). The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	Auxiliary DMA transfers are not halted.
		CLR	1	Auxiliary DMA transfers have stopped.

† For CSL implementation, use the notation PCI_PCIIS_*field_symval*

Table 40. PCI Interrupt Source Register (PCIIS) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
11	PRST	OF(*value*)		PCI reset change state bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT NOCHG	0	No change of state on PCI reset.
		CHGSTATE	1	PCI reset changed state.
10	Reserved	–	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
9	EERDY	OF(*value*)		EEPROM ready bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	EEPROM is not ready to accept a new command.
		CLR	1	EEPROM is ready to accept a new command and the data register can be read.
8	CFGERR	OF(*value*)		Configuration error bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	No checksum failure during PCI autoinitialization.
		CLR	1	Checksum failed during PCI autoinitialization. Set after an initialization due to $\overline{\text{PRST}}$ asserted and checksum error. Set after WARM if initialization has been done, but had checksum error.
7	CFGDONE	OF(*value*)		Configuration hold bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	Configuration of PCI configuration registers is not complete.
		CLR	1	Configuration of PCI configuration registers is complete. Set after an initialization due to $\overline{\text{PRST}}$ asserted. Set after WARM if initialization has been done.
6	MASTEROK	OF(*value*)		PCI master transaction completes bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	No PCI master transaction completes interrupt.
		CLR	1	PCI master transaction completes interrupt.

† For CSL implementation, use the notation PCI_PCIIS_*field_symval*

Table 40. PCI Interrupt Source Register (PCIIS) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
5	PWRHL	OF(*value*)		High-to-low transition on PWRWKP bit. The reset sources are RESET and Warm.
		DEFAULT	0	No high-to-low transition on PWRWKP.
		CLR	1	High-to-low transition on PWRWKP.
4	PWRLH	OF(*value*)		Low-to-high transition on PWRWKP bit. The reset sources are RESET and Warm.
		DEFAULT	0	No low-to-high transition on PWRWKP.
		CLR	1	Low-to-high transition on PWRWKP.
3	HOSTSW	OF(*value*)		Host software requested bit. The reset sources are RESET and Warm.
		DEFAULT	0	No host software requested interrupt.
		CLR	1	Host software requested interrupt (this bit must be set after boot from PCI to wake up DSP).
2	PCIMASTER	OF(*value*)		Master abort received bit. The reset sources are RESET and Warm.
		DEFAULT	0	No master abort received.
		CLR	1	Master abort received.
1	PCITARGET	OF(*value*)		Target abort received bit. The reset sources are RESET and Warm.
		DEFAULT	0	No target abort received.
		CLR	1	Target abort received.
0	PWRMGMT	OF(*value*)		Power management state transition bit. The reset sources are RESET and Warm.
		DEFAULT	0	No power management state transition interrupt.
		CLR	1	Power management state transition interrupt (is not set if the DSP clocks are not running).

† For CSL implementation, use the notation PCI_PCIIS_*field_symval*

15.3.4 PCI Interrupt Enable Register (PCIIEN)

The PCI interrupt enable register (PCIIEN) enables the PCI interrupts. For the DSP to monitor the interrupts, the DSP software must also set the appropriate bits in the CPU control status register (CSR) and CPU interrupt enable register (IER).

The only interrupt enabled after device reset ($\overline{\text{RESET}}$) is the HOSTSW interrupt. In this way, the PCI host can wake up the DSP by writing the DSPINT bit in the host-to-DSP control register (HDCR). The PCIIEN is shown in Figure 37 and described in Table 41.

Figure 37. PCI Interrupt Enable Register (PCIIEN)

31	16
Reserved†	
R-0	

15	13	12	11	10	9	8
Reserved†		Reserved‡	PRST	Reserved‡	EERDY	CFGERR
R-0		R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
CFGDONE	MASTEROK	PWRHL	PWRLH	HOSTSW	PCIMASTER	PCITARGET	PWRMGMT
R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0

Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.
‡ These reserved bits must always be written with 0, writing 1 to these bits result in an undefined operation.

Table 41. PCI Interrupt Enable Register (PCIIEN) Field Descriptions

Bit	***field***†	***symval***†	Value	Description
31−13	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
12	Reserved	−	0	Reserved. The reserved bit location is always read as 0. This reserved bit must always be written with a 0. Writing 1 to this bit results in an undefined operation.

† For CSL implementation, use the notation PCI_PCIIEN_*field_symval*

Table 41. PCI Interrupt Enable Register (PCIIEN) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
11	PRST	OF(*value*)		$\overline{\text{PRST}}$ transition interrupts enable bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	$\overline{\text{PRST}}$ transition interrupts are not enabled.
		DISABLE		
		ENABLE	1	$\overline{\text{PRST}}$ transition interrupts are enabled.
10	Reserved	–	0	Reserved. The reserved bit location is always read as 0. This reserved bit must always be written with a 0. Writing 1 to this bit results in an undefined operation.
9	EERDY	OF(*value*)		EEPROM ready interrupts enable bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	EEPROM ready interrupts are not enabled.
		DISABLE		
		ENABLE	1	EEPROM ready interrupts are enabled.
8	CFGERR	OF(*value*)		Configuration error interrupts enable bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	Configuration error interrupts are not enabled.
		DISABLE		
		ENABLE	1	Configuration error interrupts are enabled.
7	CFGDONE	OF(*value*)		Configuration complete interrupts enable bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	Configuration complete interrupts are not enabled.
		DISABLE		
		ENABLE	1	Configuration complete interrupts are enabled.
6	MASTEROK	OF(*value*)		PCI master transaction complete interrupts enable bit. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	PCI master transaction complete interrupts are not enabled.
		DISABLE		
		ENABLE	1	PCI master transaction complete interrupts are enabled.

† For CSL implementation, use the notation PCI_PCIIEN_*field_symval*

Table 41. PCI Interrupt Enable Register (PCIIEN) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
5	PWRHL	OF(*value*)		High-to-low PWRWKP interrupts enable bit. The reset sources are RESET and Warm.
		DEFAULT	0	High-to-low PWRWKP interrupts are not enabled.
		DISABLE		
		ENABLE	1	High-to-low PWRWKP interrupts are enabled.
4	PWRLH	OF(*value*)		Low-to-high PWRKWP interrupts enable bit. The reset sources are RESET and Warm.
		DEFAULT	0	Low-to-high PWRWKP interrupts are not enabled.
		DISABLE		
		ENABLE	1	Low-to-high PWRKWP interrupts are enabled.
3	HOSTSW	OF(*value*)		Host software requested interrupt enable bit. The reset sources are RESET and Warm.
		DISABLE	0	Host software requested interrupts are not enabled.
		DEFAULT	1	Host software requested interrupt are enabled.
		ENABLE		
2	PCIMASTER	OF(*value*)		PCI master abort interrupt enable bit. The reset sources are RESET and Warm.
		DEFAULT	0	PCI master abort interrupt is not enabled.
		DISABLE		
		ENABLE	1	PCI master abort interrupt is enabled.
1	PCITARGET	OF(*value*)		PCI target abort interrupt enable bit. The reset sources are RESET and Warm.
		DEFAULT	0	PCI target abort interrupt is not enabled.
		DISABLE		
		ENABLE	1	PCI target abort interrupt is enabled.
0	PWRMGMT	OF(*value*)		Power management state transition interrupt enable bit. The reset sources are RESET and Warm.
		DEFAULT	0	Power management state transition interrupt is not enabled.
		DISABLE		
		ENABLE	1	Power management state transition interrupt is enabled.

† For CSL implementation, use the notation PCI_PCIIEN_*field_symval*

15.3.5 DSP Master Address Register (DSPMA)

The DSP master address register (DSPMA) contains the DSP address location of the destination data for DSP master reads, or the address location of source data for DSP master writes. DSPMA also contains bits to control the address modification. DSPMA is word aligned. The DSPMA is shown in Figure 38 and described in Table 42.

Figure 38. DSP Master Address Register (DSPMA)

31	2	1	0
ADDRMA		AINC†	Rsvd‡
R/W-0		R/W-0	R-0

Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† This bit is valid on C6205 DSP only; on C64x DSP, this bit is reserved and must be written with a 0.
‡ If writing to this field, always write the default value for future device compatibility.

Table 42. DSP Master Address Register (DSPMA) Field Descriptions

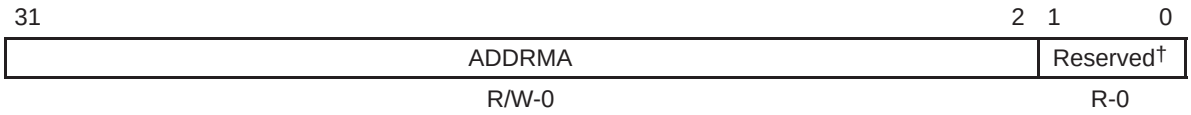
Bit	*field*†	*symval*†	Value	Description
31−2	ADDRMA	OF(*value*)	0−3FFF FFFFh	DSP word address for PCI master transactions. The reset sources are RESET and Warm.
		DEFAULT	0	ADDRMA autoincrement is enabled.
1	AINC	OF(*value*)		Autoincrement mode of DSP master address (C6205 DSP only). The reset sources are RESET and Warm. On the C64x DSP, this bit is reserved and must be written with a 0. The PCI port on the C64x DSP does not support fixed addressing for master PCI transfers. All transfers are issued to linear incrementing addresses in DSP memory. Note: On both C62x and C64x DSPs, autoincrement only affects the lower 24 bits of DSPMA. As a result, autoincrement does not cross 16M-byte boundaries and wraps around if incrementing past the boundary.
		DEFAULT	0	ADDRMA autoincrement is enabled.
		ENABLE		
		DISABLE	1	ADDRMA autoincrement is disabled.
0	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.

† For CSL implementation, use the notation PCI_DSPMA_*field_symval*

15.3.6 PCI Master Address Register (PCIMA)

The PCI master address register (PCIMA) contains the PCI word address. For DSP master reads, PCIMA contains the source address; for DSP master writes, PCIMA contains the destination address. The PCIMA is shown in Figure 39 and described in Table 43.

Figure 39. PCI Master Address Register (PCIMA)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 43. PCI Master Address Register (PCIMA) Field Descriptions

Bit	Field	*symval*†	Value	Description
31−2	ADDRMA	OF(*value*)	0−3FFF FFFFh	PCI word address for PCI master transactions. The reset sources are RESET and Warm.
		DEFAULT	0	
1−0	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.

† For CSL implementation, use the notation PCI_PCIMA_ADDRMA_*symval*

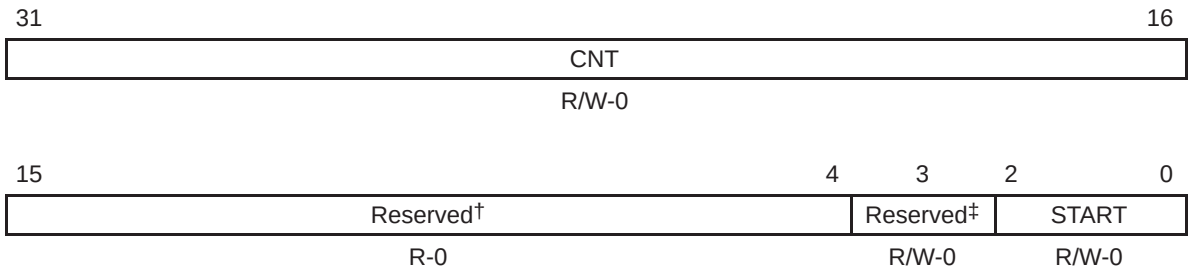
15.3.7 PCI Master Control Register (PCIMC)

The PCI master control register (PCIMC) contains:

- ❑ Start bits to initiate the transfer between the DSP and PCI.
- ❑ The transfer count, which specifies the number of bytes to transfer (64K − 1 bytes maximum).
- ❑ Reads indicate transfer status

The PCIMC is shown in Figure 40 and described in Table 44.

Figure 40. PCI Master Control Register (PCIMC)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.
‡ This reserved bit must always be written with 0, writing 1 to this bit results in an undefined operation.

Table 44. PCI Master Control Register (PCIMC) Field Descriptions

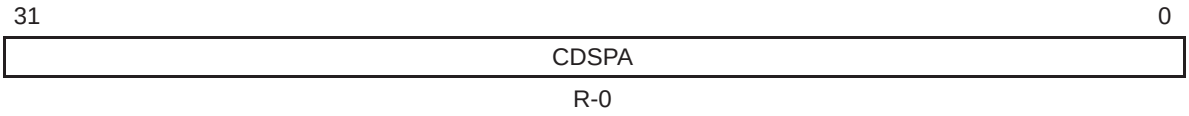
Bit	*field*†	*symval*†	Value	Description
31−16	CNT	OF(*value*)	0−FFFFh	Transfer count specifies the number of bytes to transfer. The reset sources are RESET and Warm.
		DEFAULT	0	
15−4	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
3	Reserved	−	0	Reserved. The reserved bit location is always read as 0. This reserved bit must always be written with a 0. Writing 1 to this bit results in an undefined operation.
2−0	START	OF(*value*)	0−7h	Start the read or write master transaction. The START bits return to 000b when the transaction is complete. The START bits must not be written/changed during an active master transfer. If the PCI bus is reset during a transfer, the transfer stops and the FIFOs are flushed. (A CPU interrupt can be generated on a PRST transition.) The START bits only get set if the CNT bits are not 0000h. The reset sources are RESET, PRST, and Warm.
		DEFAULT FLUSH	0	Transaction not started/flush current transaction.
		WRITE	1h	Start a master write transaction.
		READPREF	2h	Start a master read transaction to prefetchable memory.
		READNOPREF	3h	Start a master read transaction to nonprefetchable memory.
		CONFIGWRITE	4h	Start a configuration write.
		CONFIGREAD	5h	Start a configuration read.
		IOWRITE	6h	Start an I/O write.
		IOREAD	7h	Start an I/O read.

† For CSL implementation, use the notation PCI_PCIMC_*field_symval*

15.3.8 Current DSP Address Register (CDSPA)

The current DSP address register (CDSPA) contains the current DSP address for master transactions. The CDSPA is shown in Figure 41 and described in Table 45.

Figure 41. Current DSP Address (CDSPA)



Legend: R = Read only; -*n* = value after reset

Table 45. Current DSP Address (CDSPA) Field Descriptions

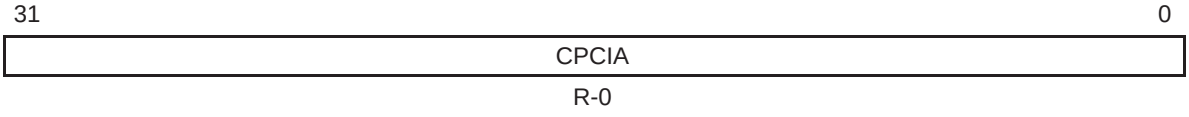
Bit	Field	*symval*†	Value	Description
31−0	CDSPA	OF(*value*)	0−FFFF FFFFh	The current DSP address for master transactions. The reset sources are RESET and Warm.
		DEFAULT	0	

† For CSL implementation, use the notation PCI_CDSPA_CDSPA_*symval*

15.3.9 Current PCI Address Register (CPCIA)

The current PCI address register (CPCIA) contains the current PCI address for master transactions. The CPCIA is shown in Figure 42 and described in Table 46.

Figure 42. Current PCI Address Register (CPCIA)



Legend: R = Read only; -*n* = value after reset

Table 46. Current PCI Address Register (CPCIA) Field Descriptions

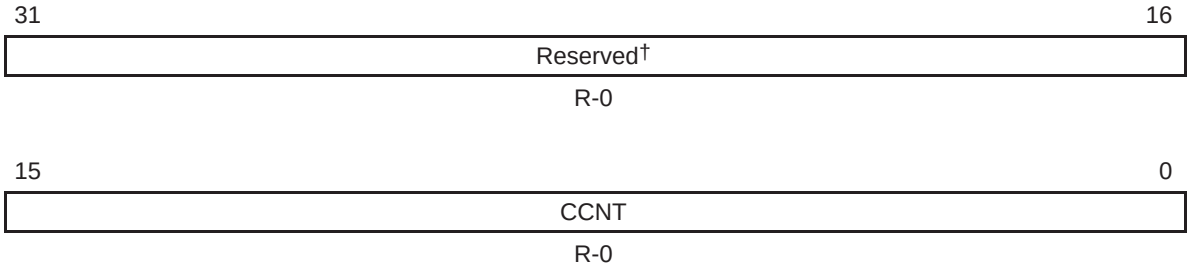
Bit	Field	*symval*†	Value	Description
31−0	CPCIA	OF(*value*)	0−FFFF FFFFh	The current PCI address for master transactions. The reset sources are RESET, PRST, and Warm.
		DEFAULT	0	

† For CSL implementation, use the notation PCI_CPCIA_CPCIA_*symval*

15.3.10 Current Byte Count Register (CCNT)

The current byte count register (CCNT) contains the number of bytes left on the current master transaction. The CCNT is shown in Figure 43 and described in Table 47.

Figure 43. Current Byte Count Register (CCNT)



Legend: R = Read only; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 47. Current Byte Count Register (CCNT) Field Descriptions

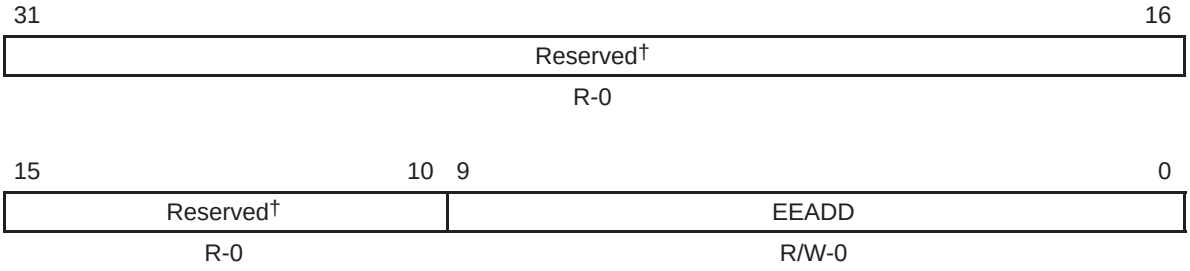
Bit	Field	*symval*†	Value	Description
31−16	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15−0	CCNT	OF(*value*)	0−FFFFh	The number of bytes left on the master transaction. The reset sources are RESET, PRST, and Warm.
		DEFAULT	0	

† For CSL implementation, use the notation PCI_CCNT_CCNT_*symval*

15.3.11 EEPROM Address Register (EEADD)

The EEPROM address register (EEADD) contains the EEPROM address. The EEADD is shown in Figure 44 and described in Table 48.

Figure 44. EEPROM Address Register (EEADD)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 48. EEPROM Address Register (EEADD) Field Descriptions

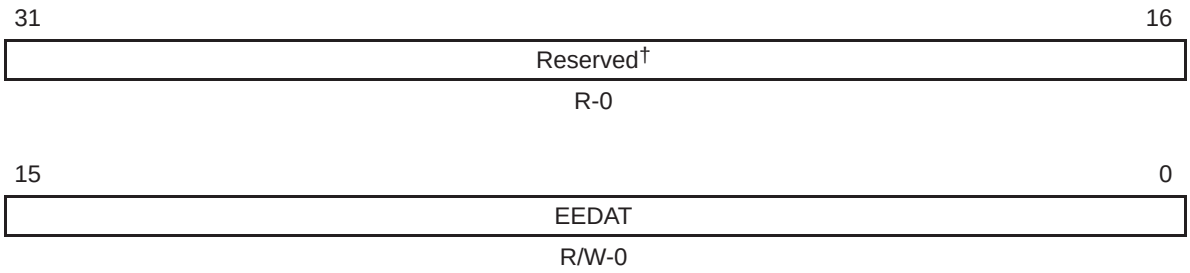
Bit	Field	*symval*†	Value	Description
31−10	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
9−0	EEADD	OF(*value*)	0−3FFh	EEPROM address. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	

† For CSL implementation, use the notation PCI_EEADD_EEADD_*symval*

15.3.12 EEPROM Data Register (EEDAT)

The EEPROM data register (EEDAT) is used to clock out user data to the EEPROM on writes and store EEPROM data on reads. For EEPROM writes, data written to EEDAT is immediately transferred to an internal register. A DSP read from EEDAT at this point does not return the value of the EEPROM data just written. The write data (stored in the internal register) is shifted out on the pins as soon as the two-bit op code is written to the EECNT bits in the EEPROM control register (EECTL). For EEPROM reads, data is available in EEDAT as soon as the READY bit in EECTL is set to 1. The EEDAT is shown in Figure 45 and described in Table 49.

Figure 45. EEPROM Data Register (EEDAT)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 49. EEPROM Data Register (EEDAT) Field Descriptions

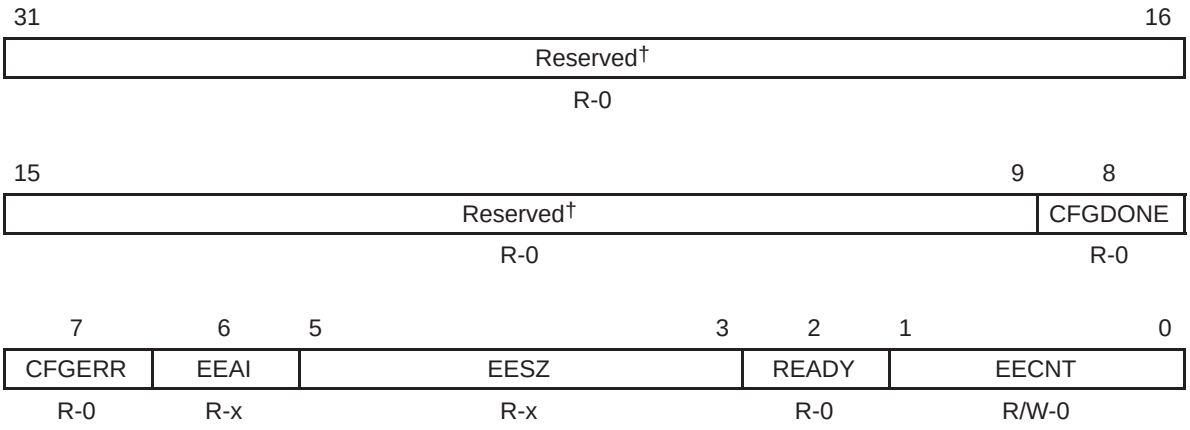
Bit	Field	*symval*†	Value	Description
31−16	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
15−0	EEDAT	OF(*value*)	0−FFFFh	EEPROM data. The reset sources are $\overline{\text{RESET}}$ and Warm.
		DEFAULT	0	

† For CSL implementation, use the notation PCI_EEDAT_EEDAT_*symval*

15.3.13 EEPROM Control Register (EECTL)

The EEPROM control register (EECTL) has fields for the two-bit opcode (EECNT) and read-only bits that indicate the size of the EEPROM (EESZ latched from the EESZ[2−0] pins on power-on reset). The READY bit in EECTL indicates when the last operation is complete, and the EEPROM is ready for a new instruction. The READY bit is cleared when a new op code is written to the EECNT bits. An interrupt can also be generated on EEPROM command completion. The EERDY bit in the PCI interrupt source register (PCIIS) and in the PCI interrupt enable register (PCIIEN) control the operation of the interrupt. The EECTL is shown in Figure 46 and described in Table 50.

Figure 46. EEPROM Control Register (EECTL)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset; -x = value is indeterminate after reset
† If writing to this field, always write the default value for future device compatibility.

Table 50. EEPROM Control Register (EECTL) Field Descriptions

Bit	*field*†	*symval*†	Value	Description
31−9	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8	CFGDONE	OF(*value*)		Configuration done bit. The reset source is $\overline{\text{RESET}}$.
		DEFAULT	0	Configuration is not done.
			1	Configuration is done.

† For CSL implementation, use the notation PCI_EECTL_*field_symval*

Table 50. EEPROM Control Register (EECTL) Field Descriptions (Continued)

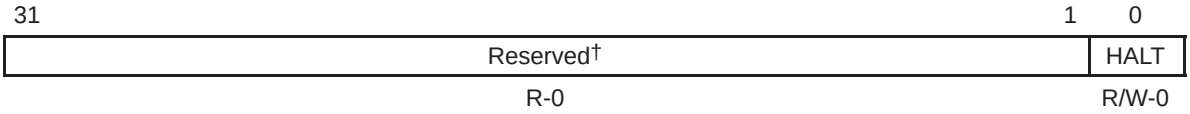
Bit	*field*†	*symval*†	Value	Description
7	CFGERR	OF(*value*)		Checksum failed error bit. The reset source is RESET.
		DEFAULT	0	No checksum error.
			1	Checksum error.
6	EEAI	OF(*value*)		EEAI pin state at power-on reset. The reset source is RESET.
		DEFAULT	0	PCI uses default values.
			1	Read PCI configuration register values from EEPROM.
5−3	EESZ	OF(*value*)	0−7h	EESZ pins state at power-on reset. The reset source is RESET.
		DEFAULT	0	No EEPROM
			1h	1K bits (C6205 DSP only)
			2h	2K bits (C6205 DSP only)
			3h	4K bits
			4h	16K bits (C6205 DSP only)
			5h−7h	Reserved
2	READY	OF(*value*)		EEPROM is ready for a new command. Cleared on writes to the EECNT bit. The reset sources are RESET and Warm.
		DEFAULT	0	EEPROM is not ready for a new command.
			1	EEPROM is ready for a new command.
1−0	EECNT	OF(*value*)	0−3h	EEPROM op code. Writes to this field cause the serial operation to commence. The reset sources are RESET and Warm.
		DEFAULT	0	Write enable (address = 11xxxx)
		EWEN		
		ERAL	0	Erases all memory locations (address = 10xxxx)
		WRAL	0	Writes all memory locations (address = 01xxxx)
		EWDS	0	Disables programming instructions (address = 00xxxx)
		WRITE	1h	Write memory at address
		READ	2h	Reads data at specified address
		ERASE	3h	Erase memory at address

† For CSL implementation, use the notation PCI_EECTL_*field_symval*

15.3.14 PCI Transfer Halt Register (HALT) (C62x DSP only)

The PCI transfer halt register (HALT) allows the C62x DSP to terminate internal transfer requests to the auxiliary DMA channel. The HALT is shown in Figure 47 and described in Table 51.

Figure 47. PCI Transfer Halt Register (HALT)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 51. PCI Transfer Halt Register (HALT) Field Descriptions

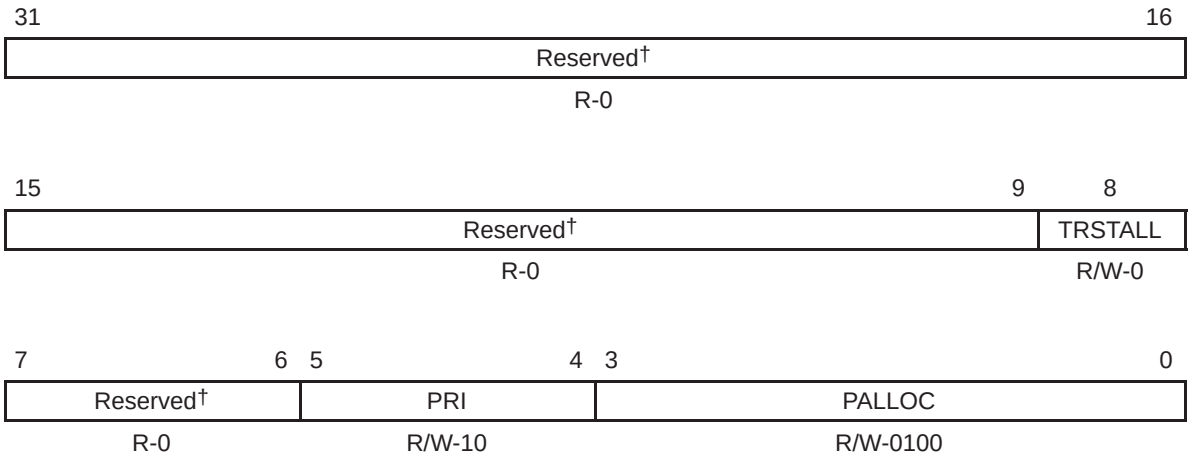
Bit	Field	*symval*†	Value	Description
31−1	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
0	HALT	OF(*value*)		Halt internal transfer requests bit. The reset sources are $\overline{\text{RESET}}$, $\overline{\text{PRST}}$, and Warm.
		DEFAULT	0	No effect.
		SET	1	HALT prevents the PCI port from performing master/slave auxiliary DMA transfer requests.

† For CSL implementation, use the notation PCI_HALT_HALT_*symval*

15.3.15 PCI Transfer Request Control Register (TRCTL) (C64x DSP only)

The PCI transfer request control register (TRCTL) controls how the PCI submits its requests to the EDMA subsystem. The TRCTL is shown in Figure 48 and described in Table 52.

Figure 48. PCI Transfer Request Control Register (TRCTL)



Legend: R = Read only; R/W = Read/Write; -*n* = value after reset
† If writing to this field, always write the default value for future device compatibility.

Table 52. PCI Transfer Request Control Register (TRCTL) Field Descriptions

Bit	***field***†	***symval***†	Value	Description
31−9	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.
8	TRSTALL	OF(*value*)		Forces the PCI to stall all PCI requests to the EDMA. This bit allows the safe changing of the PALLOC and PRI fields.
		DEFAULT	0	Allows PCI requests to be submitted to the EDMA.
			1	Halts the creation of new PCI requests to the EDMA.
7−6	Reserved	−	0	Reserved. The reserved bit location always returns the default value. A value written to this field has no effect. If writing to this field, always write the default value for future device compatibility.

† For CSL implementation, use the notation PCI_TRCTL_*field_symval*

Table 52. PCI Transfer Request Control Register (TRCTL) Field Descriptions (Continued)

Bit	*field*†	*symval*†	Value	Description
5−4	PRI	OF(*value*)	0−3h	Controls the priority queue level that PCI requests are submitted to.
			0	Urgent priority
			1h	High priority
		DEFAULT	2h	Medium priority
			3h	Low priority
3−0	PALLOC	OF(*value*)	1h−Fh	Controls the total number of outstanding requests that can be submitted by the PCI to the EDMA. Valid values of PALLOC are 1 to 15, all other values are reserved. PCI may have the programmed number of outstanding requests.
		DEFAULT	4h	Four outstanding requests can be submitted by the PCI to the EDMA.

† For CSL implementation, use the notation PCI_TRCTL_*field_symval*

Revision History

Table 53 lists the changes made since the previous version of this document.

Table 53. Document Revision History

Page	Additions/Modifications/Deletions
20	Section 3, changed 1st paragraph under Figure 4 (0180000h−0200000h) to (0180 0000h−0200 0000h).
21	Section 4, replaced 2nd paragraph.
26	Section 7.2.1, replaced 1st paragraph.
31	Section 8.3.1.1, changed subbullets under 1st bullet.
39	Section 12.3, replaced 2nd paragraph.
62	Added new Section 15.1.17. Subsequent section, figures and tables renumbered.

This page is intentionally left blank.

Index

E

F

H

I

L

M

N

O

P

R

S

T

V

W