TMS320C6452 DSP
Peripheral Component Interconnect (PCI)

User's Guide

Literature Number: SPRUF86
October 2007

Texas Instruments

Contents

List of Figures

List of Tables

Read This First

About This Manual

Describes the peripheral component interconnect (PCI) module in the TMS320C6452 Digital Signal Processor (DSP).

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.

Note: Acronyms 3PSW, CPSW, CPSW_3G, and 3pGSw are interchangeable and all refer to the 3 port gigabit switch.

TMS320C6452 DSP

Related Documents From Texas Instruments

The following documents describe the TMS320C6452 Digital Signal Processor (DSP). 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.

Data Manual—

SPRS371 —***TMS320C6452 Digital Signal Processor Data Manual*** describes the signals, specifications and electrical characteristics of the device.

CPU—

SPRU732 — ***TMS320C64x/C64x+ DSP CPU and Instruction Set Reference Guide*** describes the CPU architecture, pipeline, instruction set, and interrupts for the TMS320C64x and TMS320C64x+ digital signal processors (DSPs) of the TMS320C6000 DSP family. The C64x/C64x+ DSP generation comprises fixed-point devices in the C6000 DSP platform. The C64x+ DSP is an enhancement of the C64x DSP with added functionality and an expanded instruction set.

Reference Guides—

SPRUF85 —***TMS320C6452 DSP DDR2 Memory Controller User's Guide*** describes the DDR2 memory controller in the TMS320C6452 Digital Signal Processor (DSP). The DDR2/mDDR memory controller is used to interface with JESD79D-2A standard compliant DDR2 SDRAM devices and standard Mobile DDR SDRAM devices.

SPRUF86 — ***TMS320C6452 Peripheral Component Interconnect (PCI) User's Guide*** describes the peripheral component interconnect (PCI) port in the TMS320C6452 Digital Signal Processor (DSP). The PCI port supports connection of the C642x DSP to a PCI host via the integrated PCI master/slave bus interface. The PCI port interfaces to the DSP via the enhanced DMA (EDMA) controller. This architecture allows for both PCI master and slave transactions, while keeping the EDMA channel resources available for other applications.

SPRUF87 — ***TMS320C6452 DSP Host Port Interface (UHPI) User's Guide*** describes the host port interface (HPI) in the TMS320C6452 Digital Signal Processor (DSP). The HPI is a parallel port through which a host processor can directly access the CPU memory space. The host device functions as a master to the interface, which increases ease of access. The host and CPU can exchange information via internal or external memory. The host also has direct access to memory-mapped peripherals. Connectivity to the CPU memory space is provided through the enhanced direct memory access (EDMA) controller.

SPRUF89 — ***TMS320C6452 DSP VLYNQ Port User's Guide*** describes the VLYNQ port in the TMS320C6452 Digital Signal Processor (DSP). The VLYNQ port is a high-speed point-to-point serial interface for connecting to host processors and other VLYNQ compatible devices. It is a full-duplex serial bus where transmit and receive operations occur separately and simultaneously without interference.

SPRUF90 — ***TMS320C6452 DSP 64-Bit Timer User's Guide*** describes the operation of the 64-bit timer in the TMS320C6452 Digital Signal Processor (DSP). The timer can be configured as a general-purpose 64-bit timer, dual general-purpose 32-bit timers, or a watchdog timer.

SPRUF91 — ***TTMS320C6452 DSP Multichannel Audio Serial Port (McASP) User's Guide*** describes the multichannel audio serial port (McASP) in the TMS320C6452 Digital Signal Processor (DSP). The McASP functions as a general-purpose audio serial port optimized for the needs of multichannel audio applications. The McASP is useful for time-division multiplexed (TDM) stream, Inter-Integrated Sound (I2S) protocols, and intercomponent digital audio interface transmission (DIT).

SPRUF92 — ***TMS320C6452 DSP Serial Port Interface (SPI) User's Guide*** discusses the Serial Port Interface (SPI) in the TMS320C6452 Digital Signal Processor (DSP). This reference guide provides the specifications for a 16-bit configurable, synchronous serial peripheral interface. The SPI is a programmable-length shift register, used for high speed communication between external peripherals or other DSPs.

SPRUF93 — ***TMS320C6452 DSP Universal Asynchronous Receiver/Transmitter (UART) User's Guide*** describes the universal asynchronous receiver/transmitter (UART) peripheral in the TMS320C6452 Digital Signal Processor (DSP). The UART peripheral performs serial-to-parallel conversion on data received from a peripheral device, and parallel-to-serial conversion on data received from the CPU.

SPRUF94 — ***TMS320C6452 DSP Inter-Integrated Circuit (I2C) Module User's Guide*** describes the inter-integrated circuit (I2C) peripheral in the TMS320C6452 Digital Signal Processor (DSP). The I2C peripheral provides an interface between the DSP and other devices compliant with the I2C-bus specification and connected by way of an I2C-bus. External components attached to this 2-wire serial bus can transmit and receive up to 8-bit wide data to and from the DSP through the I2C peripheral. This document assumes the reader is familiar with the I2C-bus specification.

SPRUF95 — ***TMS320C6452 DSP General-Purpose Input/Output (GPIO) User's Guide*** describes the general-purpose input/output (GPIO) peripheral in the TMS320C6452 Digital Signal Processor (DSP). The GPIO peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs. When configured as an input, you can detect the state of the input by reading the state of an internal register. When configured as an output, you can write to an internal register to control the state driven on the output pin.

SPRUF96 —TMS320C6452 DSP Telecom Serial Interface Port (TSIP) User's Guide is a multi-link serial interface consisting of a maximum of two transmit data signals (or links), two receive data signals (or links), two frame sync input signals, and two serial clock inputs. Internally the TSIP offers single channel of timeslot data management and single DMA capability that allow individual timeslots to be selectively processed.

SPRUF97 —TMS320C6452 DSP 3 Port Switch (3PSW) Ethernet Subsystem User's Guide describes the operation of the 3 port switch (3PSW) ethernet subsystem in the TMS320C6452 Digital Signal Processor (DSP). The 3 port switch gigabit ethernet subsystem provides ethernet packet communication and can be configured as an ethernet switch. It provides the serial gigabit media independent interface (SGMII), the management data input output (MDIO) for physical layer device (PHY) management.

Peripheral Component Interconnect (PCI)

1 Introduction

This document describes the peripheral component interconnect (PCI) module in the TMS320C6452 Digital Signal Processor (DSP). The C6452 PCI is compliant to the *PCI Local Bus Specification* (revision 2.3). See that document for details on the protocol, electrical, and mechanical specifications of the PCI.

1.1 Purpose of the Peripheral

The C6452 PCI module allows communication with devices complaint to the *PCI Local Bus Specification* (revision 2.3) via a 32-bit address/data bus operating at speeds up to 66 MHZ.

1.2 Features

The PCI module supports the following features:

- *PCI Local Bus Specification* (revision 2.3) compliant
- Single function PCI interface provided
- 32-bit address/data bus width
- Operation up to 66 MHZ
- Optimized burst behavior supported for system cache line sizes of 16, 32, 64 and 128 bytes

As a slave, the PCI module includes the following features:

- Response to accesses as a 32-bit agent with medium device select ($\overline{\text{PDEVSEL}}$) timing (single wait state)
- Direct support of the Memory Read, Memory Read Multiple, Memory Read Line, Memory Write, Configuration Read and Configuration Write transactions
- Aliases Memory Write and Invalidate to the Memory Write command
- Support of variable length burst transfers up to a cache line for Memory Read Line transactions
- Support of unlimited length burst transfers for Memory Read Multiple and Memory Write transactions
- Support of single data phase transfers with disconnect for Memory Read, Configuration Read, and Configuration Write transactions
- Support of both immediate or timeout forced delayed transactions for Memory Read, Memory Read Line, and Memory Read Multiple transactions
- Support of posting of Memory Write transactions
- Support of up to six base address registers for host-to-DSP transactions
- Support of programmable cache line size of 4, 8, 16, 32, 64, or 128 bytes
- Auto-initialization sequence provided to set configuration space registers to custom values after reset

As a master, the PCI module includes the following features:

- Transaction initiation as a 32-bit agent
- Support of the Configuration Read, Configuration Write, IO Read, IO Write, Memory Read, Memory Read Line, Memory Read Multiple, Memory Write, and Memory Write and Invalidate PCI Bus commands
- Support of bursts transfers of up to 256 data phases for Memory Read Line, Memory Read Multiple, and Memory Write transactions
- Support of single data phase transfers for Memory Read transactions

- Automatic selection between Memory Read, Memory Read Line, and Memory Read Multiple based on the requested transaction length and the cache line size
- Assertion of the $\overline{\text{PIRDY}}$ signal one clock cycle after the $\overline{\text{PFRAME}}$ signal is asserted.

1.3 Features Not Supported

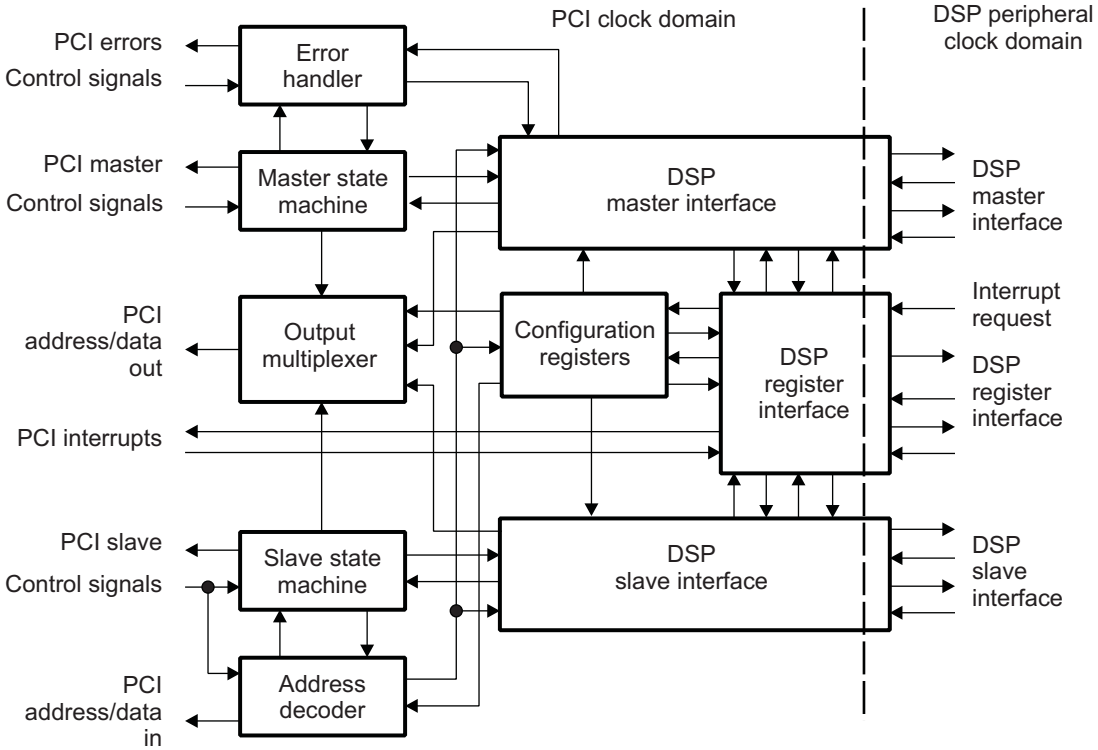
The PCI module does not support:

- PCI special cycles
- PCI interrupt acknowledge cycles
- PCI lock
- 64-bit bus operation
- Operation at frequencies greater than 66 MHZ
- Address/data stepping
- Combining (for write posting)
- Collapsing
- Merging
- Cache line wrap accesses
- Reserved accesses
- Message signaled interrupts
- Vital product data
- Slave IO Read and IO Write Transactions
- Internal Arbitration
- Power Management

1.4 Functional Block Diagram

Figure 1 shows a block diagram of the PCI module.

Figure 1. PCI Block Diagram



The PCI module consists of the following blocks:

- **Address Decoder:** This block latches transaction control information from the PCI bus and decodes that information to determine if the transaction was targeted to the PCI. This block instructs the slave state machine to either accept or ignore slave transactions as they are presented on the PCI bus.
- **Slave State Machine:** This block generates and monitors all of the PCI signals necessary for accepting transactions on the bus. All of the slave PCI protocols handling functions are split between the address decoder and slave state machine blocks.
- **DSP Slave Interface:** This block accepts transactions from the slave state machine and passes those transactions to the targeted DSP resource (for example, DDR2 memory controller). This block performs the asynchronous decoupling between the PCI clock domain and the peripheral clock domain for slave transactions. It also implements the address translation control registers and performs address translation for slave transactions.
- **DSP Master Interface:** This block accepts bus transactions from DSP masters (for example, EDMA transfer controllers) and passes those transactions on to the master state machine. It performs the asynchronous decoupling between the peripheral clock domain and the PCI clock domain for master transactions. This block implements the address translation control registers and also performs the address translation for master transactions.
- **Master State Machine:** This block generates and monitors all of the PCI signals necessary for initiating transactions on the bus. The majority of the master PCI protocols handling functions are implemented in this block. This block responds to transfer requests that are presented to it from the DSP master block.
- **Output Multiplexer:** This block multiplexes the master address, master write data, slave configuration read data, and slave memory read data on to the AD pins at the appropriate times. This block is controlled by several of the other blocks in the PCI.
- **Configuration Registers:** This block implements the required PCI configuration registers and some of the DSP registers. These registers control the modes and options in the PCI and provide vital information to the PCI host.
- **Error Handler:** This block monitors for error conditions that may occur on the PCI bus.
- **DSP Register Interface:** This block implements the asynchronous bridging function that allows DSP masters to access select PCI configuration registers, the address translation registers, and other miscellaneous PCI control/status registers. This block also implements some PCI control/status registers that reside in the peripheral clock domain.

1.5 Supported Use Case Statement

The device allows communication with devices complaint to the *PCI Local Bus Specification* (revision 2.3) via a 32-bit address/data bus operating at speeds up to 66 MHZ.

The PCI module can operate simultaneously as a PCI slave and a PCI master. As a PCI slave, the PCI module accepts configuration cycles and memory accesses from other devices. As a PCI master, the PCI module can generate configuration cycles, IO cycles, and memory accesses to other devices.

1.6 Industry Standard(s) Compliance Statement

The PCI module is compliant with the *PCI Local Bus Specification* (revision 2.3).

2 Architecture

2.1 Clock Control

The PCI module uses the following clocks:

- External PCI clock
  - Sourced by an external device through the PCLK pin.
  - Maximum clock frequency supported is 66 MHZ.
- Internal peripheral clock
  - Corresponds to SYSCLK3 of the DSP PLL Controller 1.
  - The frequency of this clock is equal to the CPU clock divided by 6.

Note: The peripheral clock frequency must not be less than the external PCI clock frequency to ensure PCI electrical timings are met.

2.2 Memory Map

The PCI can be used by an external host to access the following processor resources:

- L2 memory
- Chip level memory mapped registers
- GEM memory mapped registers
- Asyncronous EMIF memory
- DDR2 memory

Consult the *Memory Map Summary* section in the device-specific data manual for the memory address ranges of the above resources.

2.3 Signal Descriptions

Figure 2 shows the PCI signals that are used by the PCI. Table 1 shows the PCI pin name with the signal direction and description.

Figure 2. PCI Signals

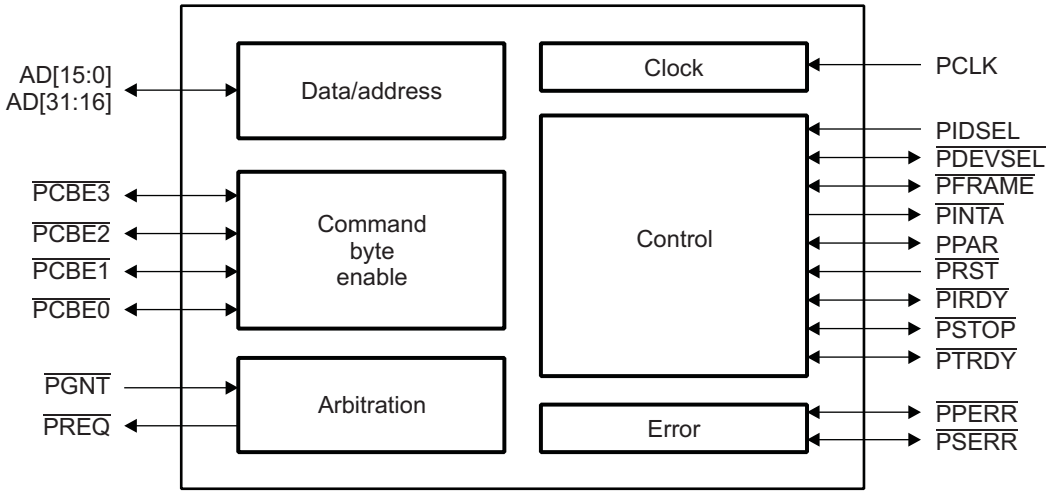


Table 1. PCI Pin Description

Pin Name	Type[1]	Description
$\overline{\text{PFRAME}}$	I/O/Z	PCI Frame
$\overline{\text{PDEVSEL}}$	I/O/Z	PCI Device Select
$\overline{\text{PSTOP}}$	I/O/Z	PCI Transaction Stop Indicator
PCLK	I	PCI Clock
$\overline{\text{PCBE}}$[3:0]	I/O/Z	PCI Command/Byte Enables
PPAR	I/O/Z	PCI Parity
$\overline{\text{PPERR}}$	I/O/Z	PCI Parity Error
$\overline{\text{PSERR}}$	I/O/Z	PCI System Error
$\overline{\text{PIRDY}}$	I/O/Z	PCI Initiator Ready
$\overline{\text{PINTA}}$	O/Z	PCI Interrupt A
$\overline{\text{PRST}}$	I	PCI Reset
PIDSEL	I	PCI Initialization Select
$\overline{\text{PTRDY}}$	I/O/Z	PCI Transmitter Ready
AD[31:16]	I/O/Z	PCI Data/Address bus [31:16]
AD[15:0]	I/O/Z	PCI Data/Address bus [16:0]
$\overline{\text{PREQ}}$	O/Z	PCI Bus Request
$\overline{\text{PGNT}}$	I	PCI Bus Grant

(1) I = Input, O = Output, Z = High impedance

2.4 Pin Multiplexing

Extensive use of pin multiplexing is used to accommodate the largest number of peripheral functions in the smallest possible package. Pin multiplexing is controlled using a combination of hardware configuration at device reset and software programmable register settings. Refer to the device-specific data sheet to determine how pin multiplexing affects the PCI module.

2.5 Byte Addressing

The PCI interface is byte-addressable. It can read and write 8-bit bytes, 16-bit half words, 24-bit words, and 32-bit words. Words are aligned on an even four-byte boundary, and 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 are fully byte-addressable, but PCI master transactions must start on a word-aligned address.

2.6 PCI Error Detection

The PCI supports the detection of the error conditions listed in Table 2. The PCI can generate an interrupt to the Host and DSP for a particular set of error conditions. The PCISTATSET, PCIHINTSET, and PCIBINTSET registers enable interrupts to the Host and DSP. The PCI also provides the status of these PCI errors, as described in section Section 2.7.

Table 2. PCI Exceptions

Exception Name	Description
PERR_DET	Data parity error. A parity error is detected when the PCI is the master during a read transaction and when the PCI is a slave during a write transaction.
SERR_DET	System error. A system error is detected when the PCI has received a target abort while mastering the bus, or when an address parity error is detected on the PCI bus.
MS_ABRT_DET	Master Abort. Generated when the PCI is a master to indicate that it terminated a master transaction with a master abort.
TGT_ABRT_DET	Target Abort. Generated when the PCI is a slave to indicate it has initiated a target abort.

2.6.1 Parity Error

If the PCI module is mastering the bus, the master data parity error (MS_DPAR_ERR) bit in the PCI command/status register (PCICSR) is set under either of the following conditions:

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

The detected parity error (DET_PAR_ERR) bit in PCICSR is set under any of the following conditions:

- If the PCI is acting as a bus master and it detects a data parity error during a read transaction.
- If the PCI is acting as a bus target and it detects a data parity error during a write transaction.
- If the PCI detects an address parity error.

The PCI will assert $\overline{\text{PPERR}}$ if the parity error response (PAR_ERR_RES) bit in PCICSR is set and the DET_PAR_ERR bit 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 the PCI, the transaction is allowed to complete unless the PCI is the master and a target disconnect is detected (that is, the PCI does not generate a master abort condition due to a parity error).

2.6.2 System Error

The PCI sets an internal system error flag under any of the following conditions:

- If an address parity error is detected on the PCI bus (even if the PCI is not the target of the transaction) and the parity error response (PAR_ERR_RES) bit in the PCI command/status register (PCICSR) is set.
- If the PCI detected $\overline{\text{PPERR}}$ asserted while mastering the bus.
- If the PCI received a target abort (disconnect without retry) while mastering the bus.

The PCI asserts $\overline{\text{PSERR}}$ if the system error pin enable (SERR_EN) bit in PCICSR is set and the internal system error flag is set. The PCI halts and waits for software or hardware reset after $\overline{\text{PSERR}}$ has been asserted. The PCI sets the signaled system error (SIG_SYS_ERR) bit in PCICSR whenever $\overline{\text{PSERR}}$ is asserted.

2.6.3 Master Abort Protocol

If a master abort occurs while the PCI is the master, the current transfer will be gracefully terminated on both the PCI bus (by de-asserting $\overline{\text{PFRAME}}$ and asserting $\overline{\text{PIRDY}}$) and the peripheral bus (by supplying ready signals through the DSP master interface until the burst is completed). Both the received master abort (RCV_MS_ABRT) bit in the PCI command/status register (PCICSR) and the received master abort (RCV_MS_ABRT) bit in the PCI command/status mirror register (PCICSRMIR) are set.

2.6.4 Target Abort Protocol

If a target abort occurs while the PCI is the master, the current transfer will be gracefully terminated on both the PCI bus and the peripheral bus (in the same way as for the master abort). Both the received target abort (RCV_TGT_ABRT) bit in the PCI command/status register (PCICSR) and the received target abort (RCV_TGT_ABRT) bit in the PCI command/status mirror register (PCICSRMIR) are set.

2.6.5 Retry/Disconnect Protocol

If a transaction is disconnected or retried, the master will unconditionally repeat the transaction starting at the location of the first remaining uncompleted word. The DSP has no knowledge of retry or disconnections on the bus.

2.7 Status Reporting

The PCI module provides the status of various PCI errors generated or detected by it in the command/status register (PCICSR) and an internal status register.

The PCICSR is in the PCI configuration register space. An external host can access this register through the TYPE0 configuration space access. The DSP can access this register through the command/status mirror register (PCICSRMIR).

The PCICSR provides the status of the following error conditions:

- Detected Parity Error (DET_PAR_ERR)
- Signaled System Error (SIG_SYS_ERR)
- Received Master Abort Error (RCV_MS_ABRT)
- Received Target Abort Error (RCV_TGT_ABRT)
- Signaled Target Abort (SIG_TGT_ABRT)
- Master Data Parity Reported (MS_DPAR_REP)

Status bits in PCICSR cannot be set manually. They are set only by the PCI module. A status bit in this register can be cleared by writing a 1 to that bit.

The internal status register provides the status of the following PCI interrupt and errors:

- Software interrupts (SOFT_INT)
- Parity Error Detected (PERR_DET)
- System Error Detected (SERR_DET)
- Master Abort Error Detected (MS_ABRT_DET)
- Target Abort Error Detected (TGT_ABRT_DET)

The PCI provides the status set register (PCISTATSET) and the status clear register (PCISTATCLR) to set and clear the bits in the internal status register. Reading of both these registers returns the value of the internal status register. The internal status register is internal to the PCI module and is not directly accessible to an external host. It is also not directly accessible to the DSP.

To clear an error bit in the command/status register the corresponding bit in the internal status register also needs to be cleared first. The bits in the internal status register can be cleared through the status clear register (PCISTATCLR).

Setting or clearing a bit in the internal status register does not affect the corresponding bit in the command/status register. Similarly, clearing the command/status register does not affect the corresponding bit in the internal status register.

An interrupt can be generated to an external host (through the $\overline{\text{PINTA}}$ pin) and to the DSP through the bits of the internal status register provided the corresponding bit is enabled in the internal host/DSP interrupt enable register. See Section 2.9 for more details on interrupt support.

2.8 Reset Considerations

2.8.1 PCI Reset Sources

The device has both an internal and external reset source (see Figure 3).

The external reset is asserted through either the PCI reset pin ($\overline{\text{PRST}}$) or the device power-on reset pin ($\overline{\text{POR}}$). The $\overline{\text{PRST}}$ pin is the main PCI hardware reset. This resets most of the PCI logic within the external PCI clock domain. This reset brings PCI-specific registers, sequencers, and signals to a consistent state. The $\overline{\text{POR}}$ pin is one of the main device hardware reset pins. This reset affects the entire device, not just the PCI module.

The internal reset is controlled by the power and sleep controller (PSC). The PSC asserts the internal reset of the PCI module after a device-level global reset or when the DSP code programs the PSC to do so. A device-level global reset is generated through the power-on reset pin ($\overline{\text{POR}}$) or the warm reset pin ($\overline{\text{RESET}}$); however, there are other methods to generate a device-level global reset. See Table 3 for these methods and use conditions for different reset sources when using the PCI on the device.

Figure 3. Internal and External Reset Inputs of the PCI Module

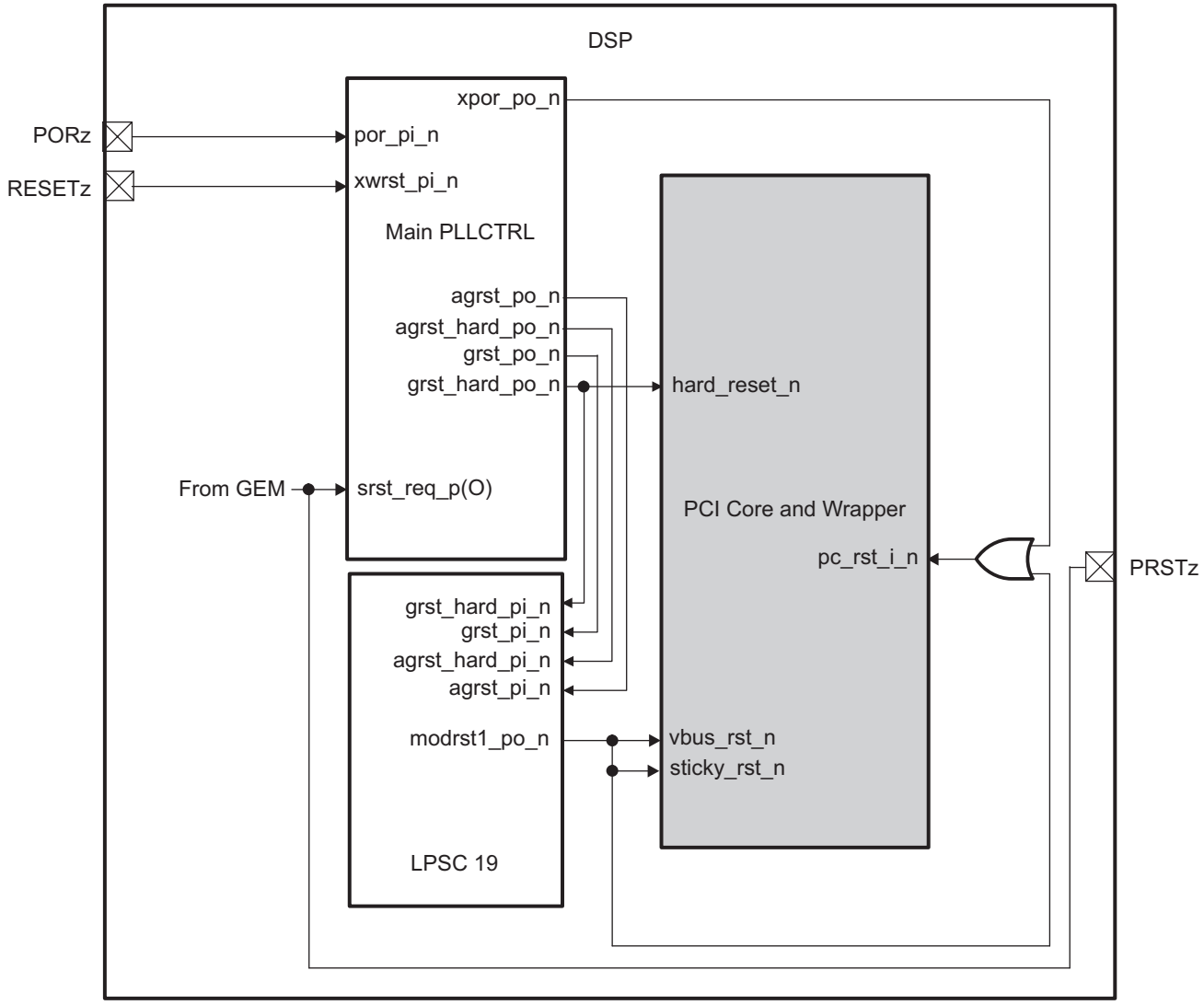


Table 3. Device-Level Global Reset Sources When Using the PCI Module

Reset Name	Description
Power on Reset through $\overline{\text{PORz}}$ Pin	• Entire chip is reset during a power-on reset • The external reset on the PCI is asserted internally and the PSC asserts the internal PCI reset until programmed
Warm reset generated through $\overline{\text{RESET}}$ Pin	• Entire chip minus on-chip emulation logic is reset • The PSC asserts the internal PCI reset until programmed
System Reset generated through $\overline{\text{PRSTz}}$ Pin	• Entire chip minus on-chip emulation logic is reset • The PSC asserts the internal PCI reset until programmed • The PSC should be programmed to only de-assert the internal PCI reset after the external PCI reset $\overline{\text{PRSTz}}$ has been asserted.

Table 3. Device-Level Global Reset Sources When Using the PCI Module (continued)

Reset Name	Description
PSC Reset	• DSP code configures the PSC of the PCI to assert the internal PCI reset. • The PSC of the PCI should not be used to place the PCI in reset after it has taken the PCI out of reset.

2.8.2 PCI Register Reset Values

Some of the PCI memory-mapped registers (Section 5.2) are connected to configuration hook registers. See Table 9 for the list of configuration hook registers supported. The values in these configuration hook registers are latched to the actual PCI registers on PCI reset. The default values in these configuration hook registers can be overwritten by software. These registers are implemented mainly to support PCI I2C EEPROM auto-initialization, as discussed in Section 2.12.3.

2.9 Interrupt Support

The PCI can generate an interrupt for the status conditions listed in Table 4. When a status condition is set, the PCI can raise an interrupt to the DSP or PCI host or both based on what interrupts have been enabled for host and DSP. See Section 2.9.1 and Section 2.9.2 for enabling interrupts to host and DSP for a particular set of status conditions.

Table 4. PCI Interrupts

Interrupt Name	Description
Parity Error	A parity error is detected when the PCI is the master during a read transaction and when the PCI is a slave during a write transaction.
System Error	A system error is detected when the PCI has received a target abort while mastering the bus, or when an address parity error is detected on the PCI bus.
Master Abort	Generated when the PCI is a master to indicate that it terminated a master transaction with a master abort.
Target Abort	Generated when the PCI is a slave to indicate it has initiated a target abort.

2.9.1 DSP-to-Host Interrupts

The PCI can raise an interrupt to an external host for various status conditions, as described in Table 4. The PCI includes an internal host interrupt enable register that specifies which status conditions will generate interrupts to a host. The PCI host interrupt enable register is not directly accessible by a host or the DSP. Two registers are provided to set or clear bits in the internal host interrupt enable register: the host interrupt enable set register (PCIHINTSET) and the host interrupt enable clear register (PCIHINTCLR).

An interrupt for a particular status condition can be enabled by setting the corresponding bit in PCIHINTSET. Interrupt generation through the $\overline{\text{PINTA}}$ pin is enabled by default when PCI is in D0 power state. Reading PCIHINTSET returns the contents of the internal host interrupt enable register.

An interrupt for a particular status condition can be disabled by setting the corresponding bit in PCIHINTCLR. Reading PCIHINTCLR returns the bitwise ANDing of the internal status register and the internal host interrupt enable register. PCIHINTCLR is typically read by the host to determine the source of an interrupt when $\overline{\text{PINTA}}$ is asserted.

A level-sensitive active-low interrupt is generated to an external host on $\overline{\text{PINTA}}$ if a bit in the internal status register is asserted and the corresponding bit in the internal host interrupt enable register is also asserted. When an interrupt is raised on $\overline{\text{PINTA}}$, a host can read PCIHINTCLR to determine the status condition that caused the interrupt. The host can indirectly access PCIHINTCLR through the base address registers.

Software can also use the SOFT_INT bits to interrupt a host via $\overline{\text{PINTA}}$. Software interrupts are enabled and disabled by writing to the host interrupt enable register. Setting the corresponding bit in the internal status register will assert a level sensitive active low interrupt on $\overline{\text{PINTA}}$. The DSP or a host can clear this interrupt condition by clearing applicable bit in the internal status register.

2.9.2 Host-to-DSP Interrupts

The PCI can raise an interrupt to the DSP for various status conditions described in Table 4. The PCI includes an internal DSP interrupt enable register that specifies which status conditions will generate interrupts to the DSP. The DSP interrupt enable register is not directly accessible by an external host or the DSP. Two registers are provided to set or clear bits in the internal DSP interrupt enable register: the DSP interrupt enable set register (PCIDINTSET) and the DSP interrupt enable clear register (PCIDINTCLR).

An interrupt for a particular status condition can be enabled by setting the corresponding bit in PCIDINTSET. Reading PCIDINTSET returns the contents of the internal DSP interrupt enable register.

An interrupt for a particular status condition can be disabled by setting the corresponding bit in PCIDINTCLR. Reading PCIDINTCLR returns the bitwise ANDing of the internal PCI status register and the internal DSP interrupt enable register. PCIDINTCLR is typically read by the DSP to determine the source of an interrupt.

An interrupt request is generated to the DSP if a bit in the internal status register (PCISTATSET) is asserted and the corresponding bit in the internal DSP interrupt enable register (PCIDINTSET) is also asserted. When an interrupt is raised to the DSP, the DSP can read PCIDINTCLR to know the status condition that caused the interrupt.

The interrupt request to the DSP is generated through the CPU PCI interrupt (PCIINT). This interrupt can also be forced by setting the SOFT_INT bits in the status set register (PCISTATSET).

2.9.3 Interrupt Multiplexing

The PCI has a single interrupt source (PCIINT) to the DSP CPU. This interrupt source is not multiplexed with any other interrupt on the CPU.

2.10 DMA Event Support

The PCI module does not generate any EDMA events.

2.11 Emulation Considerations

The PCI is not directly affected by emulation accesses. However, please note that other resources which the PCI has access to can be affected by emulation accesses.

2.12 PCI Configuration

The operation of PCI is configured through the PCI configuration registers and memory-mapped registers.

2.12.1 Programming the PCI Configuration Registers

The PCI configuration registers can be programmed by an external host and the DSP. Normally, a host programs a part of configuration registers and DSP programs the remaining part.

A host can control modes and options in PCI by programming the configuration registers. For example, a host can program the command/status register (PCICSR) to enable PCI bus master capability and to enable the memory access to DSP. It can program the base address registers (PCIBAR0 to PCIBAR5) to map the DSP memory regions into the PCI address space.

The DSP needs to program a set of configuration registers before the PCI host system software scans the PCI, as part of enumerating the PCI devices. For example, it needs to program the vendor ID/device ID register (PCIVENDEV) and the class code/revision ID register (PCICLREV) so that the PCI host system software can identify the device and load the respective host driver if required. This can be done automatically using the I2C EEPROM auto-initialization method, as described in Section 2.12.3.

A host can access the configuration registers by performing a TYPE 0 access in the PCI bus. The DSP cannot access the configuration registers directly. To facilitate access to the configuration registers, mirror registers are defined in the PCI memory-mapped register space. Updating the mirror registers updates the corresponding configuration registers.

2.12.2 Programming the PCI Memory-Mapped Registers

The memory-mapped registers include configuration mirror registers, master and slave address translation registers, and other miscellaneous control registers. Memory-mapped registers can be programmed by an external host and the DSP.

A host can access the memory-mapped registers through the slave interface supported by the PCI, provided the DSP has mapped those registers to the PCI memory space through the slave window base address registers. The DSP may directly program the configuration mirror registers, slave and master address translation registers, and other miscellaneous configuration registers.

2.12.3 PCI I2C EEPROM Auto-Initialization

The PCI can be configured using the default values in the PCI Module registers or a select set of registers may be programmed via an EEPROM. The device treats any PCI boot as an internal ROM boot. The device has two separate boot modes to distinguish if it's a PCI boot with or without auto-initialization—i.e. if the PCI defaults need to be modified. If PCI Boot Mode With Auto-Initialization is selected (BOOTMODE[3:0] = 0010,PCI I2C EEPROM Auto-Initialization), the boot-loader software will copy some PCI parameters from an I2C ROM, and store them in the PCI Wrapper MMRs. The values from these PCI Wrapper MMRs will be latched by PCI as defaults to the PCI's MMRs at PCI reset de-assertion (pc_rst_i_n). Meanwhile, because CONFIG_DONE=0 (indicating Configuration Registers can be being loaded), PCI interface does not accept accesses. Once the initialization is complete, the sw will pulse the CONFIG_DONE bit in the PCI Wrapper allowing PCI interface to accept accesses.

This will be the boot-loader software sequence to accommodate the above PCI peripheral requirements:

- Take the PCI out of reset through PSC– CONFIG_DONE should be 0 at this point.
  - If it's PCI boot without auto-initialization, skip this step.
- If it's PCI boot with auto-initialization
  - Initialize both the PCI back end registers and PCI Wrapper registers
  - Enable the I2C through PSC
  - Read the default values from I2C
- Set CONFIG_DONE=1 so that PCI interface can accept accesses

If auto-initialization is not enabled, the PCI configuration registers are left with their default values and the I2C EEPROM is not accessed for PCI configuration purposes. The values in the PCI Wrapper registers will not get latched into the PCI back end registers until the next time the host resets the PCI peripheral through the PRSTn pin (which may not occur until much later in operation). This is not an issue because the boot-loader is initializing both the PCI Wrapper registers and the PCI Back End Registers directly. As far as the external host is concerned, it doesn't know when the boot-loader is done. The external host may begin transaction as soon as it is out of reset (or it may also check RESETOUTn to see that the device is internally released from reset). During a PCI transaction, after the PCI arbiter grants access of the bus to the host, they host will assert FRAME and IRDY and wait for the device to assert PTRDY and DEVSEL to continue the transfer. If the boot-loader hasn't set CONFIG_DONE=1, the device doesn't give PTRDY and DEVSEL. As a result the host will restart the transfer. As soon as the boot-loader is done (CONFIG_DONE=1), the host will eventually get PTRDY. Table 5 lists the default values for some of the PCI configuration registers. These default values can be changed by enabling PCI I2C EEPROM auto-initialization.

Table 5. PCI Configuration Register Default Values

PCI Configuration Register	Default Value
Vendor ID	104Ch
Device ID	B003h
Revision ID	01h
Class Code	11 8000h
Subsystem Vendor ID	0000h
Subsystem ID	0000h
Interrupt Pin	00h
Interrupt Line	00h
Minimum Grant	00h
Maximum Latency	00h

2.12.3.1 PCI Auto-Initialization from I2C EEPROM

When auto-initialization is used, the PCI configuration registers are programmed by the on-chip ROM Boot Loader (RBL) with the values stored in an I2C EEPROM.

PCI I2C EEPROM auto-initialization is enabled when BOOTMODE[3:0] = 0010b, PCIEN = 1, and FASTBOOT = 1. If auto-initialization is not enabled, the PCI configuration registers are left with their default values and the I2C EEPROM is not accessed for PCI configuration purposes. The function of the BOOTMODE[3:0], PCIEN, and FASTBOOT pins is fully described in the device data manual, refer to that document for more details.

When auto-initialization is enabled, the CONFIG_DONE bit in the configuration done register (PCICFGDONE) takes a default value of 0. This prevents the PCI from responding to any requests. When auto-initialization is completed, the RBL sets the CONFIG_DONE bit to 1 to allow the PCI to respond to requests.

2.12.3.2 I2C EEPROM Memory Map

The on-chip ROM Boot Loader requires big-endian format for the data stored in the I2C EEPROM. Byte addresses 400h through 41Bh of the I2C EEPROM are reserved for auto-initialization of PCI configuration registers. The remaining locations are not used for auto-initialization and can be used for storing other data. Table 6 summarizes the I2C EEPROM memory layout, as required for PCI auto-initialization.

Table 6. I2C EEPROM Memory Layout

Byte Address	Contents
400h	Vendor ID [15:8]
401h	Vendor ID [7:0]
402h	Device ID [15:8]
403h	Device ID [7:0]
404h	Class code [7:0]
405h	Revision ID [7:0]
406h	Class code [23:16]
407h	Class code [15:8]
408h	Subsystem vendor ID [15:8]
409h	Subsystem vendor ID [7:0]
40Ah	Subsystem ID [15:8]
40Bh	Subsystem ID [7:0]
40Ch	Max_Latency
40Dh	Min_Grant

Table 6. I2C EEPROM Memory Layout (continued)

Byte Address	Contents
40Eh-418h	Reserved (use 00h)
419h	Checksum [15:8]
41Ah	Checksum [7:0]

2.12.3.3 I2C EEPROM Checksum

The PCI configuration data contained in the I2C EEPROM is checked against a checksum. The configuration data bytes are treated as an array of 16-bit words (little-endian format). The checksum is a 16-bit cumulative exclusive-OR (XOR) of the configuration data words, starting with an initial value of AAAAh. You must ensure that the proper 16-bit checksum value is written to address 419h and 41Ah when programming the I2C EEPROM.

Checksum = AAAAh XOR Word0(401h:400h) XOR Word1(403h:402)… XOR Word12(418h:417h)

If the I2C EEPROM is not accessed for PCI configuration purposes (that is, PCI_EEAI = 0 at reset), then the checksum is not performed. If the checksum fails, the on-chip ROM bootloader defaults to the UART boot and it does not set the CONFIG_DONE bit in the configuration done register (PCICFGDONE).

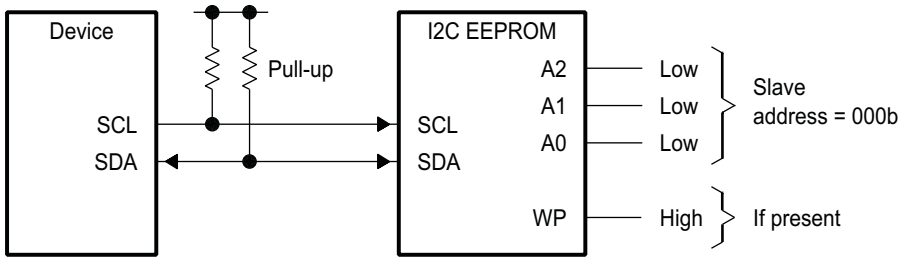
2.12.3.4 DSP I2C EEPROM Interface

For PCI auto-initialization, the DSP supports I2C EEPROMs or devices operating as I2C slaves with the following features:

- The memory device complies with Philips I2C Bus Specification v 2.1
- The memory device uses two bytes for internal addressing; that is, the read/write bit followed by two bytes for addressing
- The memory device has the capability to auto-increment its internal address counter such that the contents of the memory device can be read sequentially

During PCI auto-initialization, the DSP acts as the master and the I2C EEPROM acts as the slave. Figure 4 shows the minimum connection required between the DSP and one I2C EEPROM. The required pull-ups must be placed on SDA and SCL to ensure that the I2C EEPROM interface works correctly. The slave address of the I2C EEPROM slave address must be set to 50h.

Figure 4. Signal Connections for I2C EEPROM Boot Mode



Some I2C EEPROMs have a write-protect (WP) feature that prevents unauthorized writes to memory. This feature is not needed for auto-initialization because the DSP will only read data from the I2C EEPROM. The write protect feature can be enabled or disabled.

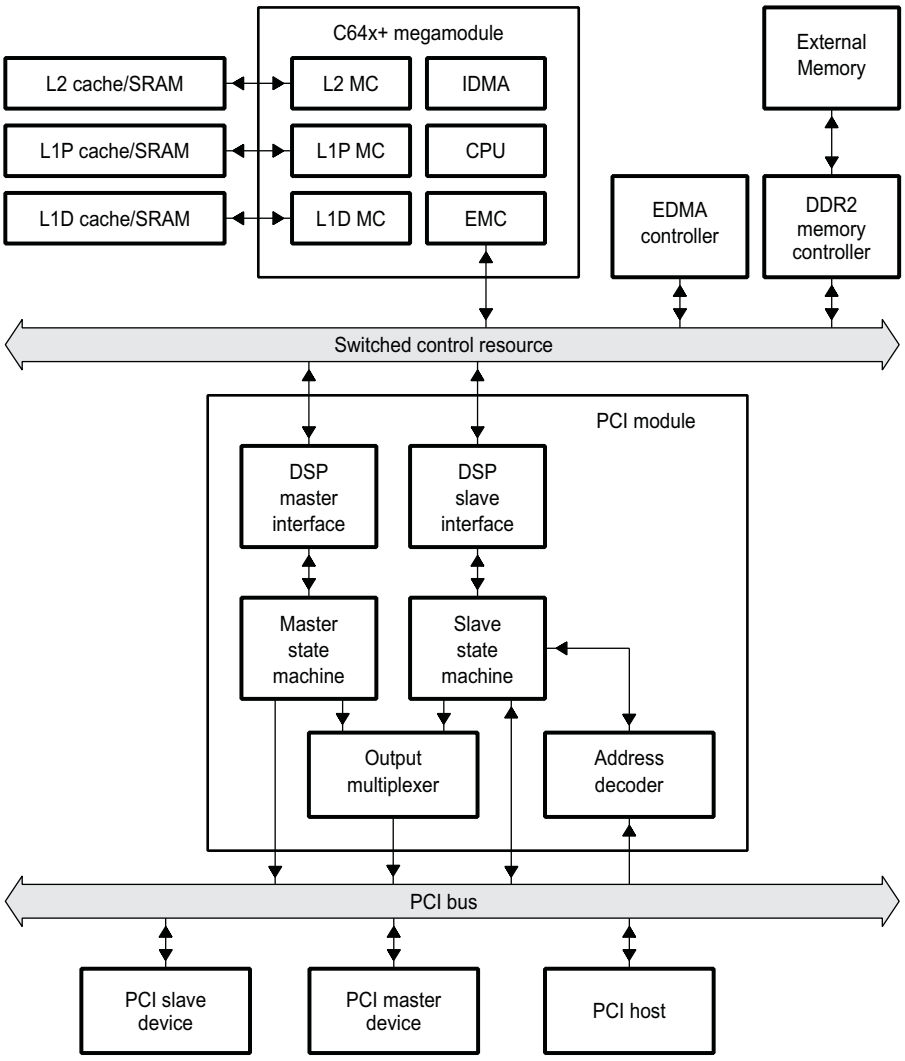
For PCI auto-initialization purposes only byte address 400-401h are used. The remaining locations in the I2C EEPROM can be used for other purposes.

For detailed information on the I2C, see the Inter-Integrated Circuit (I2C) Module User's Guide.

2.13 Connecting a Local PCI to an External PCI Device

Figure 5 shows a simplified block diagram of how the PCI module interfaces local DSP master modules (EDMA controller, CPU, etc.) and other DSP resources (DDR2 memory controller, DSP internal memory, etc.) to external PCI memory and external PCI masters.

Figure 5. PCI to External PCI Device



(1) EDMA: Enhanced Direct Memory Access Controller
EMC: Extended Memory Controller
L1P MC: L1 P Memory Controller
L1D MC: L1D Memory Controller
L2 MC: L1 Memory Controller
IDMA: Internal Direct Memory Access Controller

The following steps show how the PCI module interfaces local DSP master modules (EDMA controller, CPU, etc.) to external PCI memory:

1. A DSP master initiates a transaction aimed at external PCI memory through the DSP switched central resource.
2. The address is decoded by the DSP master interface.
3. The DSP master interface claims the transaction if the DSP address falls within the master memory map (described in Section 4.1).
4. DSP master interface translates the DSP address into a PCI address and generates a request to the master state machine.
5. The master state machine initiates a transaction on the PCI bus using the PCI address.
6. The request is received by the external PCI host, which responds accordingly.

The following steps show how the PCI module interfaces external PCI masters to DSP resources (DDR2 memory controller, DSP internal memory, etc.):

1. External PCI master initiates a transaction on the PCI bus.
2. The address decoder decodes the PCI address of the transaction and instructs the PCI slave state machine to claim the transaction if the PCI address falls within the slave memory map (described in Section 3.1) assigned to the DSP.
3. The slave state machine forwards the request to the DSP slave interface.
4. The DSP slave interface translates the PCI address into a DSP address and places the DSP address on the DSP switched central resource.
5. All DSP slaves decode the address to determine if they are being accessed. If so, they respond accordingly. For example, in the case of an external memory access, the DDR2 memory controller accesses external memory using the DSP address.

3 PCI Slave Operation

The PCI slave operates in response to transfer requests that are presented on the PCI bus. The PCI slave was intended to enable high performance read and write performance through the use of delayed transactions combined with prefetching for reads and posting for writes. The PCI slave supports two FIFOs/buffers (a read and write) for efficient data transfer. Each buffer holds 16 32-bit words of read or write data.

3.1 Slave Memory Map

The PCI module provides full visibility for an external host into DSP memory through six sets of PCI slave base address translation registers (PCIBAR0TRL, PCIBAR1TRL, PCIBAR2TRL, PCIBAR3TRL, PCIBAR4TRL, and PCIBAR5TRL) and PCI base address mask registers (PCIBAR0MSK, PCIBAR1MSK, PCIBAR2MSK, PCIBAR3MSK, PCIBAR4MSK, and PCIBAR5MSK). The DSP can use any of these sets of registers to map any DSP memory region to the PCI memory map. These registers can be configured by software at any time. The default values of these registers provide the mapping shown in Table 7.

Section 3.2 and Section 3.3 explain how to map a region in the PCI host address space to a region in the DSP memory space by setting up a slave window.

Table 7. PCI Base Addresses

Base Address	Window Size	Prefetchable	Memory Space
0	8MB	Yes	L2 RAM memory space
1	4MB	No	GEM MMRs
2	4MB	No	Chip-level MMRs
3	8MB	Yes	EMIF3B CE2 memory space
4	8MB	Yes	EMIF3B CE3 memory space
5	8MB	Yes	DDR2 CE0 memory space

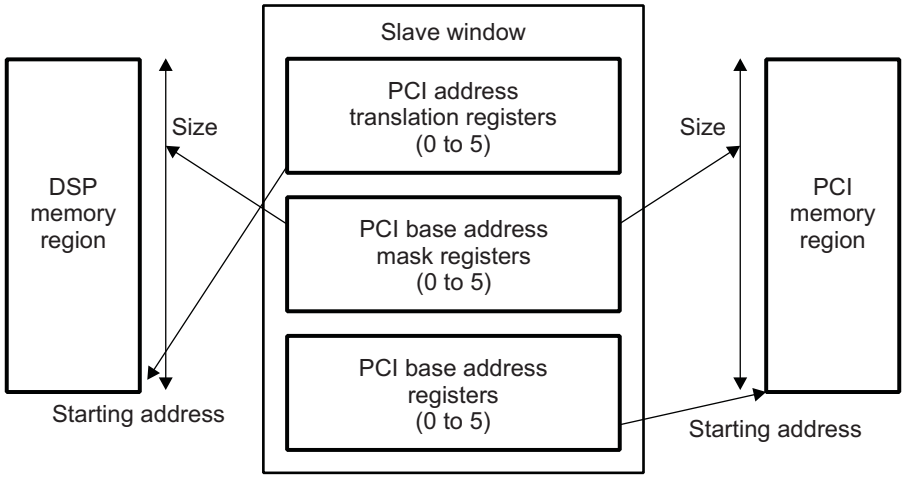
3.2 Configuring Slave Window Registers

A slave window maps a region in the DSP memory space to a region in the PCI address space. This allows a PCI host to access the DSP memory through the PCI address space. A slave window is configured with the following registers:

- PCI slave base address translation register: Configures the starting address of the window in the DSP address space.
- PCI base address register: Configures the starting address of the slave window in the PCI address space.
- PCI base address mask register: Configures the size of the window and prefetchability of the DSP memory region being mapped.

PCI supports six slave window configurations with the support of these registers. For more information on slave access address translation, see Section 3.3. A slave window configuration is shown in Figure 6.

Figure 6. Slave Window Configuration



3.2.1 Configuration of Base Address Registers 0 to 5 (PCIBAR*n*) by PCI Host

The base address registers (PCIBAR*n*) allow an external host to map the DSP's address space into the host memory or I/O address space.

The base address registers reside in PCI configuration space and a host normally configures them. A host can access base address registers 0 to 5 (PCIBAR0, PCIBAR1, PCIBAR2, PCIBAR3, PCIBAR4 and PCIBAR5) by performing a TYPE 0 access in the PCI bus.

The base address registers contain the following bit fields:

- ADDR (31-4): These bits specify the base address of the slave window on the PCI address space.
- PREFETCH (3): This bit specifies the prefetchablity of the memory space controlled by the base address register.
- TYPE (2-1): These bits specify whether the base address maps to PCI I/O address space or memory address space. The device supports only mapping into PCI memory space.
- IOMEM_SP_IND (0): The size of the base address register, either 32 or 64 bits. The device only supports 32-bit addressing.

Normally, a host configures the ADDR bits of the base address registers during its boot time when it enumerates all the PCI devices. The write-access of a host to each of the ADDR bits is determined by the corresponding bit in the address mask (ADDRMASK) bits of the base address mask registers (PCIBAR*n*MSK). A bit in ADDR is read-only to a host when its corresponding bit in ADDRMASK is cleared. Conversely, a bit in ADDR can be both read and written by a host when its corresponding bit in ADDRMASK is set. The DSP is required to complete the configuration of the base address mask registers before the host attempts to configure the base address registers.

3.2.2 Configuration of Slave Base Address Translation Registers 0 to 5 (PCIBAR*n*TRL) by DSP

A slave base address translation register (PCIBAR*n*TRL) configures the DSP side parameters of a slave window. There are six slave base address translation registers (PCIBAR0TRL, PCIBAR1TRL, PCIBAR2TRL, PCIBAR3TRL, PCIBAR4TRL, and PCIBAR5TRL) that allow six slave windows to be set up. The slave base address translation registers are usually configured by the DSP; however, they can also be programmed by an external host. PCI slave base address translation registers control the translation of transaction addresses as they flow from the external PCI bus to the DSP. Section 3.3 explains the translation of PCI addresses to DSP addresses.

3.2.3 Configuration of Base Address Mask Registers 0 to 5 (PCIBAR*n*MSK) by DSP

A base address mask register (PCIBAR*n*MSK) configures the size and prefetchablity of a slave window. There are six slave base address translation registers (PCIBAR0MSK, PCIBAR1MSK, PCIBAR2MSK, PCIBAR3MSK, PCIBAR4MSK, and PCIBAR5MSK) available in the PCI to support six slave windows. The base address mask registers are usually configured by the DSP, however they can also be programmed by an external host. The DSP can access the base address mask registers directly, as they are mapped to the DSP memory space.

The base address mask registers contain the following bit fields:

- ADDRMASK: These bits control the PCI host write access of the corresponding bits in the PCI configuration base address registers (0 to 5).
- PREFETCH_EN: This bit specifies whether or not the memory space controlled by the corresponding configuration base address register is prefetchable. This bit is reflected in bit 3 of the corresponding configuration base address register.

The DSP is required to complete the configuration of the base address mask registers before the host attempts to configure the base address registers.

3.3 Slave Access Address Translations

Window configurations control the translation of transaction addresses as they flow from the external PCI bus to the DSP. This translation process uses the contents of the corresponding base address mask register (PCIBAR0MSK, PCIBAR1MSK, PCIBAR2MSK, PCIBAR3MSK, PCIBAR4MSK, and PCIBAR5MSK) to determine which of the bits in the PCI address should be modified. Bits 31-4 (ADDRMASK) are replaced in the address where the corresponding bit in the base address mask register is set by the corresponding bit in the slave base address translation register (PCIBAR*n*TRL).

The following steps occur during a PCI-to-DSP address translation:

1. During the address phase, an external PCI master places the PCI address on the address bus AD[31-0].
2. The PCI finds the appropriate slave window for the address by comparing the address bits given on AD[31-*n*] with the corresponding bits in the base address register of all the slave windows one by one. The value of *n* is the number of bits set in the corresponding base address mask register of the slave window. The minimum value of *n* is 4. The value of *n* indicates the number of significant bits in the PCI address that needs to be decoded. If the address on AD[31-*n*] matches the corresponding bits of the base address register of any one of the slave windows, the PCI claims the PCI transaction. Otherwise, it ignores the transaction.
3. If the PCI claims the transaction, it generates a DSP address by replacing bits 31-*n* in the PCI address with the corresponding bits in the base address translation register of the previously selected slave window.

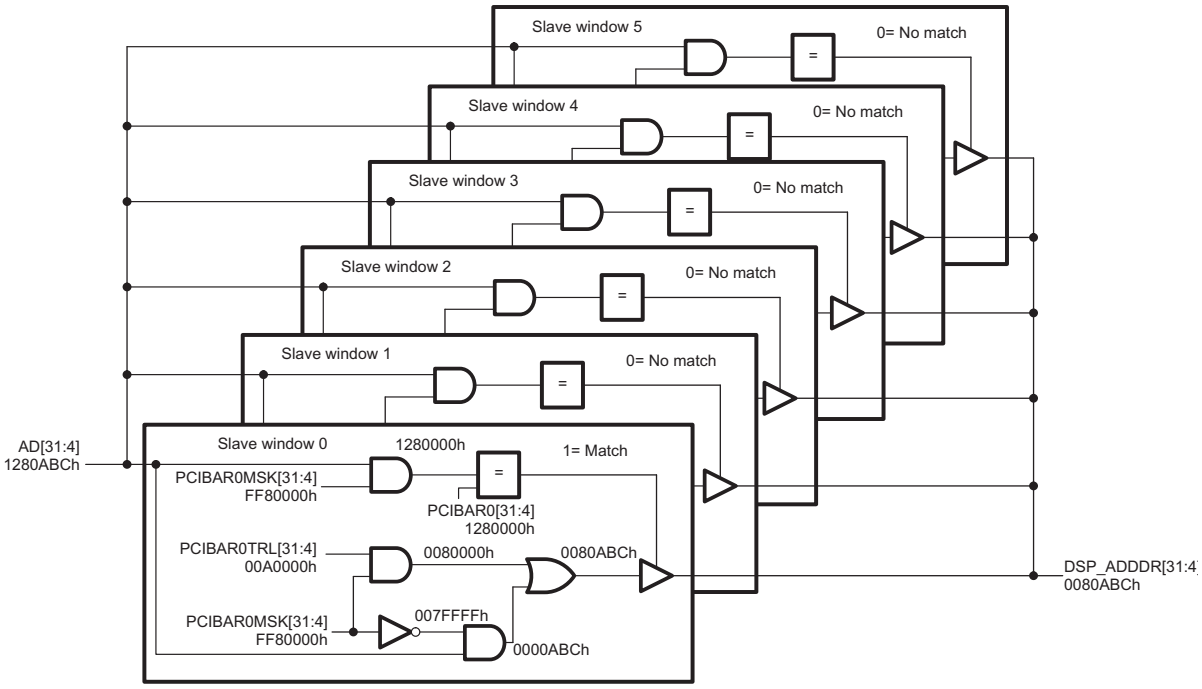
Figure 7 gives an example of a PCI-to-DSP address translation using a PCI address of 1280 ABC0h. In this example, slave window 0 is created using this configuration:

PCIBAR0 = 1280 0000h, PCIBAR0MSK = FF80 0008h, and PCIBAR0TRL = 00A00 000h.

With these settings, slave window 0 translates PCI addresses from 1280 0000h to 12FF FFFFh (8 MB) to DSP addresses 0080 0000h to 00FF FFFFh. The following is the sequence of events for generating the DSP address:

1. The external PCI master places the PCI address 1280 ABC0h on the address bus AD[31-0].
2. The PCI finds the slave window corresponding to the PCI address by comparing the address given on AD[31-4], 128 0ABCh, with the corresponding bits in the PCI base address registers. The PCI base address mask registers indicate bits in the PCI address that need to be compared.
3. Slave window 0 has PCIBAR0[31-4] set to 128 0000h and PCIBAR0MSK[31-4] = FF8 0000h. Therefore, the PCI matches the PCI address with slave window 0 and claims transaction.
4. The value of the PCIBAR0MSK[31-4] bits is inverted and ANDed with the PCI address AD[31-4]. The PCIBAR0TRL[31-4] bits are also ANDed with the value of the PCIBAR0MSK[31-4] bits. The resulting values are ORed to form the DSP address (0080 ABCh).

Figure 7. PCI-to-DSP Address Translation



3.4 Slave Configuration Operations

3.4.1 Configuration Write Transactions

The decoding of and response to a configuration write transaction by the PCI depends on the following:

- PCBE[3-0] must be Bh during the address phase.
- PIDSEL must be asserted during the address phase.
- AD[1-0] must be 00b during the address phase.

If the above conditions are met, the transaction is decoded as a hit and the PCI will assert $\overline{\text{PDEVSEL}}$ using medium decode timing. $\overline{\text{PTRDY}}$ will be asserted coincident with the assertion of $\overline{\text{PDEVSEL}}$. If the master on the PCI bus intends to perform more than a single data phase transaction (as determined by the state of $\overline{\text{PFRAME}}$), $\overline{\text{PSTOP}}$ will also be asserted coincident with the assertion of $\overline{\text{PDEVSEL}}$ and $\overline{\text{PTRDY}}$ to signal a disconnect. $\overline{\text{PSTOP}}$ will continue to be asserted until $\overline{\text{PFRAME}}$ is de-asserted and $\overline{\text{PIRDY}}$ is asserted in accordance with the PCI specification.

Configuration write transactions will never result in a retry. Because the configuration registers are included within the PCI, no transactions will occur on any of the DSP slave interface as a result of a configuration write transaction.

3.4.2 Configuration Read Transactions

Decoding of and response to a configuration read transaction by the PCI depends on the following:

- PCBE[3-0] must be Ah during the address phase.
- PIDSEL must be asserted during the address phase.
- AD[1-0] must be 00b during the address phase.

If the above conditions are met, the transaction is decoded as a hit and the PCI slave will assert

PDEVSEL using medium decode timing. PTRDY will be asserted one cycle following the assertion of PDEVSEL. If the master on the PCI bus intends to perform more than a single data phase transaction (as determined by the state of PFRAME), PSTOP will also be asserted coincident with the assertion of PTRDY to signal disconnect. PSTOP will continue to be asserted until PFRAME is de-asserted and PIRDY is asserted in accordance with the PCI specification.

Configuration read transactions will never result in a retry. Because the configuration registers are included within the PCI, no transactions will occur on any of the PCI blocks as a result of a configuration read transaction.

3.5 Slave Memory Operations

3.5.1 Memory Write or Memory Write and Invalidate Transactions

The PCI treats memory write and memory write and invalidate transactions identically and is therefore not intended to support cacheable memory spaces.

The decoding of and response to a memory write or a memory write and invalidate transaction by the PCI depends on the following conditions:

- PCBE[3-0] must be 7h (memory write) or Fh (memory write and invalidate) during the address phase.
- At least one of the six base address registers (PCIBAR0, PCIBAR1, PCIBAR2, PCIBAR3, PCIBAR4, and PCIBAR5) must have bit 0 (IOMEM_SP_IND) cleared to 0.
- The address that is given on AD[31-*n*] during the address phase must match the value in the corresponding bits of one of the memory base address registers. The value of *n* is based on the base address mask registers (PCIBAR0MSK, PCIBAR1MSK, PCIBAR2MSK, PCIBAR3MSK, PCIBAR4MSK, and PCIBAR5MSK) and indicates the number of significant bits in the address to be decoded. The minimum value of *n* is 4.
- The PCI slave write buffer is empty.
- The address that is given on AD[1-0] during the address phase must be 00b (linear addressing).

If only the first three of the previous conditions are met, a retry will be issued because the PCI slave write buffer is not empty. If only the first four conditions are met, a single data phase transfer will be completed on the PCI bus. This transfer will be identical in behavior to a configuration write transaction. If the byte enables have at least one byte asserted, a write transaction will initiate on the DSP as soon as any pending read burst requests have completed. If no byte enables are asserted, the transaction will be terminated internally in the PCI and no DSP transactions will occur. If all of the above conditions are met, a multi-data phase transfer will be completed. No wait states will be inserted by the PCI via the target ready indicator (PTRDY), but wait states inserted by the master will be properly handled. The burst will be allowed to continue until one of the following conditions occurs:

- The PCI slave write buffer is almost full.
- A data phase with no asserted byte enables is encountered.
- The burst is about to extend beyond the address boundary of the PCI.
- The master ends the transaction.

3.5.2 Memory Read Transactions

The decoding of and response to a memory read transaction by the PCI depends on the following conditions:

- PCBE[3-0] must be 6h during the address phase.
- At least one of the six base address registers (PCIBAR0, PCIBAR1, PCIBAR2, PCIBAR3, PCIBAR4, and PCIBAR5) must have bit 0 (IOMEM_SP_IND) cleared to 0.
- The address on AD[31-*n*] during the address phase must match the value in the corresponding bits of one of the memory base address registers. The value of *n* is based on the base address mask registers (PCIBAR0MSK, PCIBAR1MSK, PCIBAR2MSK, PCIBAR3MSK, PCIBAR4MSK, and PCIBAR5MSK) and indicates the number of significant bits in the address to be decoded. The minimum value of *n* is 4.
- A delayed read is not outstanding, or this transaction is a re-request of a pending delayed read request

and the read data is available. A delayed read is a transaction that must complete on the destination bus before completing on the originating bus.

- The PCI slave write buffer is empty.

If only the first three of the previous conditions are met, a retry will be issued because the delayed transaction is not ready to complete or the PCI slave write buffer is not empty. If the first four or all of the conditions are met, a single data phase transfer will be completed on the PCI bus. This transfer will be similar in behavior to a configuration transaction.

If the transaction is not a delayed completion and the byte enables have at least one byte asserted, a single word read transaction will initiate on the DSP as soon as any pending write burst requests have completed on the interface. If no byte enables are asserted and the addressed memory region is not prefetchable, the transaction will be terminated internally in the PCI and no DSP transactions will occur. The byte enables which were presented for the first data phase on the PCI bus will be inverted and presented as the byte enables for the transfer. The PCI can wait up to 12 PCLK cycles for data to be returned from the DSP slave interface. If data does not return within this time, a retry will be issued and the transaction will be tagged as a delayed read. Alternatively, for performance reasons, if the FORCE_DEL_READ bit in the slave control register (PCISLVCNTL) is asserted, a retry and delayed read can be immediately forced without waiting for the 12 PCLK cycles.

If the transaction is a delayed completion and the data is available, the transaction will complete immediately with $\overline{\text{PTRDY}}$ being asserted coincident with $\overline{\text{PDEVSEL}}$. Only a single-word prefetch is supported for the memory read command even when it is used within prefetchable memory regions.

3.5.3 Memory Read Line Transactions

The decoding of and response to a memory read line transaction by the PCI depends on the following conditions:

- PCBE[3-0] must be Eh during the address phase.
- At least one of the six base address registers (PCIBAR0, PCIBAR1, PCIBAR2, PCIBAR3, PCIBAR4, and PCIBAR5) must have bit 0 (IOMEM_SP_IND) cleared to 0.
- The address on AD[31-*n*] during the address phase must match the value in the corresponding bits of one of the memory base address registers. The value of *n* is based on the base address mask registers (PCIBAR0MSK, PCIBAR1MSK, PCIBAR2MSK, PCIBAR3MSK, PCIBAR4MSK, and PCIBAR5MSK) and indicates the number of significant bits in the address to be decoded. The minimum value of *n* is 4.
- A delayed read is not outstanding, or this transaction is a re-request of a pending delayed read line request and the read data is available. A delayed read is a transaction that must complete on the destination bus before completing on the originating bus.
- The PCI slave write buffer is empty.

If only the first three previous conditions are met, a retry will be issued because the delayed transaction is not ready to complete or the PCI slave buffer is not empty. If the first four or all of the conditions are met, a burst read operation will be initiated on the DSP as soon as any pending write burst requests have completed on the interface. The length of the transfer will be the number of words from the requested address to the end of the cache line, unless this value exceeds the size of the PCI slave read data FIFO (16 words). If the burst size is larger than the PCI slave read data FIFO, the transfer will be broken up into 8 word transfers in the same way as the memory read multiple transfers. Since memory read line transactions are prefetchable, all byte enables are asserted internally during the burst. The PCI can wait up to 12 PCLK cycles for data to be returned from the DSP slave interface. If data does not return within this time, a retry will be issued and the transaction will be tagged as a delayed read. Alternatively, for performance reasons, if the FORCE_DEL_READ_LN bit in the slave control register (PCISLVCNTL) is asserted, a retry and delayed read can be immediately forced without waiting for the 12 PCLK cycles. If the transaction is a delayed completion and the data is available, the first data phase of the transaction will complete immediately with $\overline{\text{PTRDY}}$ being asserted coincident with $\overline{\text{PDEVSEL}}$. $\overline{\text{PTRDY}}$ will continue to be asserted until all of the prefetched data has been read or the burst is terminated by the master. If the master attempts to burst beyond the current cache line, the PCI will assert $\overline{\text{PSTOP}}$ on the next to the last data phase in the cache line.

3.5.4 Memory Read Multiple Transactions

The decoding of and response to a memory read multiple transaction by the PCI depends on the following conditions:

- PCBE[3-0] must be Ch during the address phase.
- At least one of the six base address registers (PCIBAR0, PCIBAR1, PCIBAR2, PCIBAR3, PCIBAR4, and PCIBAR5) must have bit 0 (IOMEM_SP_IND) cleared to 0.
- The address on AD[31-*n*] during the address phase must match the value in the corresponding bits of one of the memory base address registers. The value of *n* is based on the base address mask registers (PCIBAR0MSK, PCIBAR1MSK, PCIBAR2MSK, PCIBAR3MSK, PCIBAR4MSK, and PCIBAR5MSK) and indicates the number of significant bits in the address to be decoded. The minimum value of *n* is 4.
- A delayed read is not outstanding, or this transaction is a re-request of a pending delayed read line request and the read data is available. A delayed read is a transaction that must complete on the destination bus before completing on the originating bus.
- The PCI slave write buffer is empty.

If only the first three previous conditions are met, a retry will be issued because the delayed transaction is not ready to complete or the PCI slave buffer is not empty. If the first four or all of the conditions are met, a burst read operation will be initiated on the DSP as soon as any pending write burst request have completed on the interface. The length of the transfer will be 16 words for the initial transfer of a burst. As memory read multiple transactions are prefetchable, all byte enables are asserted internally during the burst. The PCI can wait up to 12 PCLK cycles for data to be returned from the DSP slave interface. If data does not return within this time, a retry will be issued and the transaction will be tagged as a delayed read. Alternatively, for performance reasons, if the FORCE_DEL_READ_MUL bit in the slave control register (PCISLVCNTL) is asserted, a retry and delayed read can be immediately forced without waiting for the 12 PCLK cycles.

If the transaction is a delayed completion and the data is available, the first data phase of the transaction will complete immediately with $\overline{\text{PTRDY}}$ being asserted coincident with $\overline{\text{PDEVSEL}}$. $\overline{\text{PTRDY}}$ will continue to be asserted as long as data is available in the read prefetch buffer or the burst is terminated by the master. As data is transferred from the read prefetch buffer on to the PCI bus, additional 8 word read fetches will be performed on the DSP slave interface as buffer space becomes available.

4 PCI Master Operation

The PCI operates as a master in response to memory transfer requests that are presented on the DSP master interface, and to indirect IO and configuration requests that are presented via the master configuration/IO transaction proxy registers. For memory transactions, the PCI can be programmed to only use the basic memory read or write transactions, or can also perform burst transfers as efficiently as possible by automatically selecting the proper command for memory transactions based on the transaction length and the cache line size. The PCI supports two FIFOs/buffers (a read and write) for efficient data transfer. Each buffer holds 16 32-bit words of read or write data.

4.1 Master Memory Map

The PCI enables the DSP to access the PCI memory through the master memory map. There is 256MB of space dedicated for PCI memory in the DSP memory map. This 256MB space is divided into 32 windows of 8MB fixed size, see Table 8. These windows are called master windows or PCI address windows. Each master window can be configured individually to map 8MB of PCI memory to the DSP address space.

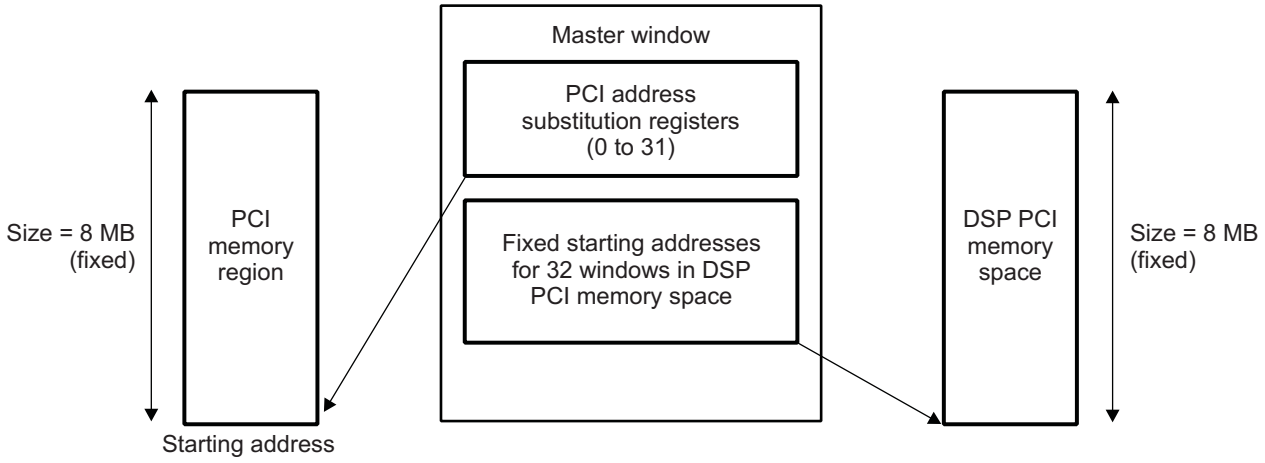
Table 8. PCI Master Windows

Master Window Number	Base Address Register	Window Size	DSP Memory Range
0	PCIADDSUB0	8MB	4000 0000h-407F FFFFh
1	PCIADDSUB1	8MB	4080 0000h-40FF FFFFh
2	PCIADDSUB2	8MB	4100 0000h-417F FFFFh
3	PCIADDSUB3	8MB	4180 0000h-41FF FFFFh
4	PCIADDSUB4	8MB	4200 0000h-427F FFFFh
5	PCIADDSUB5	8MB	4280 0000h-42FF FFFFh
6	PCIADDSUB6	8MB	4300 0000h-437F FFFFh
7	PCIADDSUB7	8MB	4380 0000h-43FF FFFFh
8	PCIADDSUB8	8MB	4400 0000h-447F FFFFh
9	PCIADDSUB9	8MB	4480 0000h-44FF FFFFh
10	PCIADDSUB10	8MB	4500 0000h-457F FFFFh
11	PCIADDSUB11	8MB	4580 0000h-45FF FFFFh
12	PCIADDSUB12	8MB	4600 0000h-467F FFFFh
13	PCIADDSUB13	8MB	4680 0000h-46FF FFFFh
14	PCIADDSUB14	8MB	4700 0000h-477F FFFFh
15	PCIADDSUB15	8MB	4780 0000h-47FF FFFFh
16	PCIADDSUB16	8MB	4800 0000h-487F FFFFh
17	PCIADDSUB17	8MB	4880 0000h-48FF FFFFh
18	PCIADDSUB18	8MB	4900 0000h-497F FFFFh
19	PCIADDSUB19	8MB	4980 0000h-49FF FFFFh
20	PCIADDSUB20	8MB	4A00 0000h-4A7F FFFFh
21	PCIADDSUB21	8MB	4A80 0000h-4AFF FFFFh
22	PCIADDSUB22	8MB	4B00 0000h-4B7F FFFFh
23	PCIADDSUB23	8MB	4B80 0000h-4BFF FFFFh
24	PCIADDSUB24	8MB	4C00 0000h-4C7F FFFFh
25	PCIADDSUB25	8MB	4C80 0000h-4CFF FFFFh
26	PCIADDSUB26	8MB	4D00 0000h-4D7F FFFFh
27	PCIADDSUB27	8MB	4D80 0000h-4DFF FFFFh
28	PCIADDSUB28	8MB	4E00 0000h-4E7F FFFFh
29	PCIADDSUB29	8MB	4E80 0000h-4EFF FFFFh
30	PCIADDSUB30	8MB	4F00 0000h-4F7F FFFFh
31	PCIADDSUB31	8MB	4F80 0000h-4FFF FFFFh

4.2 Configuring Master Windows

Each master window corresponds to 8 MB of the DSP's PCI memory address space. For example, window 0 corresponds to DSP addresses 4000 000h–407F FFFFh, and window 1 corresponds to the next 8 MB, 4080 0000h–40FF FFFFh. Each window can map an 8 MB of PCI memory to its corresponding DSP's PCI memory address space through its address substitution register. Figure 8 displays a master window configuration.

Figure 8. Master Window Configuration



There are 32 address substitution registers (PCIADDSUB*n*) available in the PCI. Each of these registers corresponds to a master window. These registers reside in the PCI interface and are normally programmed by the DSP. Figure 9 shows an example of an address substitution register.

Bits 31-23 (ADD_SUBS) of the register contain the MSBs of the PCI addresses within the corresponding window. The remaining reserved 23 bits are the size of the window and these bits have no function. Reads of this field will return 0s. Section 4.3 explains the translation of addresses as a transaction flow from the DSP domain to the PCI domain.

Figure 9. PCI Address Substitution Register (0 to 31)

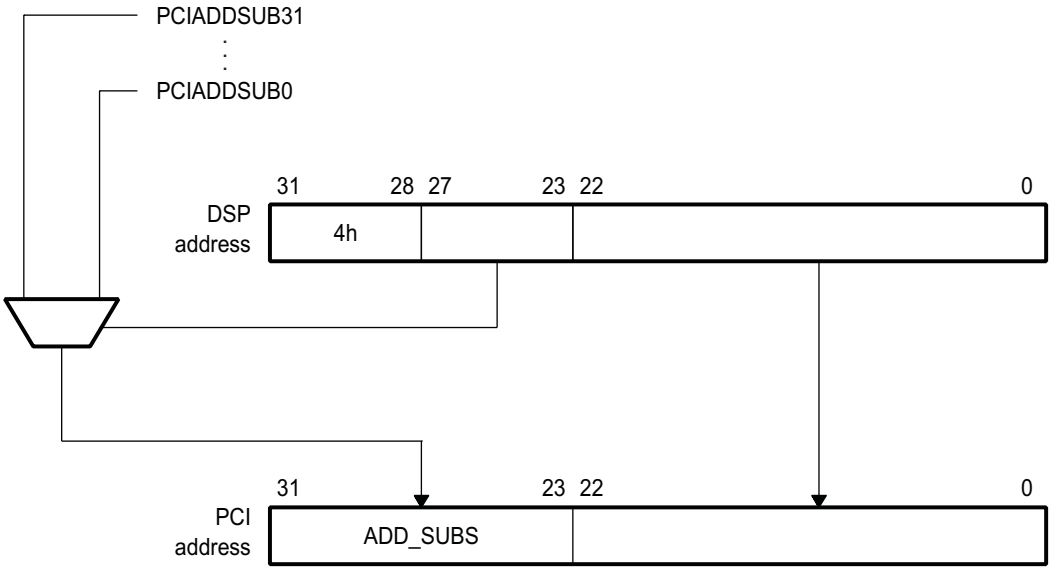
31	24	23	0
ADD_SUBS		Reserved	
R/W-0		R-0	

LEGEND: R/W = Read/Write, R = Read only; -*n* = value after reset

4.3 Master Address Translation

Address translation from the DSP to the PCI domain is done using the address substitution register (PCIADDSUB*n*). Figure 10 shows the address just prior to the address translation.

Figure 10. DSP-to-PCI Address Translation



During the address translation, the upper 4 bits are don't cares, as they have already been used for the address decode to reach the DSP's PCI memory space. The next 5 bits decide which PCI address window corresponds to the DSP address. Once the window is determined, the 23 LSBs of the DSP address are allowed to pass through to the PCI address and the upper 9 bits are taken from the ADD_SUBS field of the corresponding PCIADDSUB*n*. Figure 11 illustrates this process.

Figure 11. Example of DSP-to-PCI Address Translation

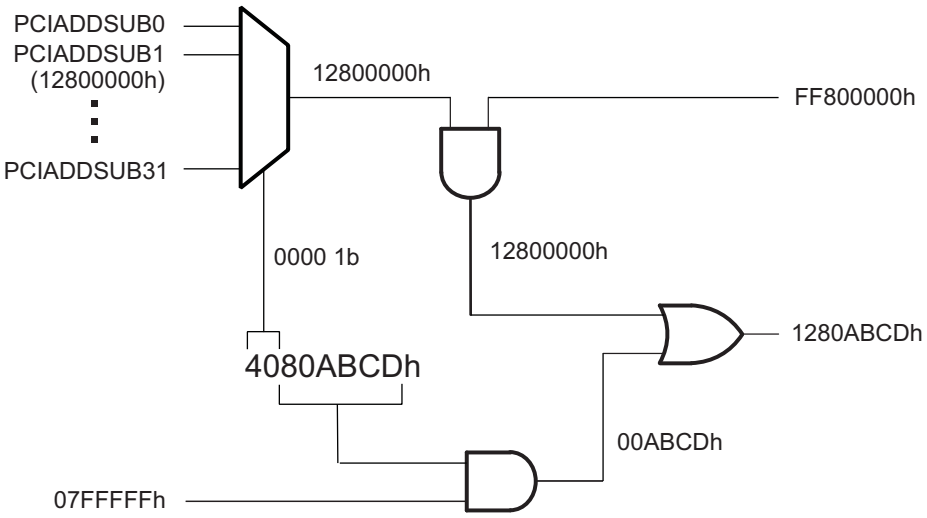


Figure 11 shows an example of a DSP-to-PCI address translation. In this example, the DSP address is 4080 ABCDh and PCIADDSUB1 is set to 1280 0000h.

1. The 4 MSBs of the address (4h) indicate the DSP's PCI memory space is being accessed.
2. The next 5 bits (00001b) determine that the destination is the second 8MB PCI window, signifying that PCIADDSUB1 will be used. Note that bits 27-3 directly correspond to the PCIADDSUB*n* used.
3. The upper 9 bits of the address substitution register 1 and the lower 23 bits of the DSP address are concatenated to form the PCI address (1280 ABCDh).
4. This address is used by the PCI module.

4.4 Master Configuration Operations

During a configuration space access, AD[31-2] is used to address the PCI device, the function within the device and a DWORD in the function's configuration space. AD[1-0] is ignored. However, you can set AD[1-0] to 00b for TYPE 0 access or set AD[1-0] to 01b for TYPE 1 access.

The byte enables select the bytes within the addressed DWORD. The byte enables allow access to a byte, word, DWORD, or non-contiguous bytes in the addressed DWORD. In the addressed DWORD, BE0 enables byte 0, BE1 enables the byte 1, and so on.

4.4.1 Configuration Write Transactions

The DSP causes the PCI to perform a configuration write operation by executing the following steps:

1. Reading the master configuration/IO access command register (PCIMCFGCMD) and ensuring that the READY bit (bit 31) is asserted.
2. Writing the data for the configuration write to the master configuration/IO access data register (PCIMCFGDAT) through the DSP register interface.
3. Writing the address for the configuration write to the master configuration/IO access address register (PCIMCFGADR).
4. Writing to the master configuration/IO access command register (PCIMCFGCMD) with the TYPE field cleared to 0 (configuration transaction), the RD_WR field cleared to 0 (write), and the BYTE_EN fields set to the desired value.

On the next cycle, $\overline{\text{PREQ}}$ is asserted. Once $\overline{\text{PGNT}}$ is sampled asserted, the PCI asserts $\overline{\text{PFRAME}}$ and outputs the write address onto the AD pins and the configuration write command (Bh) onto the PCBE pins. On the next cycle, the master asserts $\overline{\text{PIRDY}}$, de-asserts $\overline{\text{PFRAME}}$, and outputs the data to be written. The PCI will never insert wait states during a transfer but will respond to wait states as controlled by $\overline{\text{PTRDY}}$. As is required, the PCI continually checks the bus for exceptions (master abort, target abort, retry, disconnect, latency timeout, parity error, system error) while the transfer is ongoing. A ready signal is returned to the DSP master interface through the READY bit in the master configuration/IO access command register (PCIMCFGCMD) when the transfer is complete.

4.4.2 Configuration Read Transactions

The DSP can request that the PCI to perform a configuration read operation by doing the following:

1. Reading the master configuration/IO access command register (PCIMCFGCMD) and ensuring that the READY bit (bit 31) is asserted.
2. Writing the address for the configuration write to the master configuration/IO access address register (PCIMCFGADR).
3. Writing to the master configuration/IO access command register (PCIMCFGCMD) with the TYPE field cleared to 0 (configuration transaction), the RD_WR field set to 1 (read), and the BYTE_EN fields set to the desired value.

On the next cycle, PREQ is asserted. Once PGNT is sampled asserted, the PCI asserts PFRAME and outputs the read address onto the AD pins and the configuration read command (Ah) onto the PCBE pins. On the next cycle, the PCI asserts PIRDY, and de-asserts PFRAME. The PCI will never insert wait states during a transfer but will respond to wait states as controlled by PTRDY. As is required, the master continually checks the bus for exceptions (master abort, target abort, retry, disconnect, latency timeout, parity error, system error) while the transfer is ongoing. When the transfer is complete, the data is returned to the master configuration/IO access data register (PCIMCFGDAT) and the READY bit in the master configuration/IO access command register (PCIMCFGCMD) is set to 1.

4.5 Master I/O Operations

In the I/O address space, all 32 AD lines are used to provide a full byte address. The PCI that initiates an I/O transaction is required to ensure that AD[1-0] indicates the least significant valid byte for the transaction.

The byte enables indicate the size of the transfer and the affected bytes within the DWORD and must be consistent with AD[1-0]. Table 9 lists the valid combinations for AD[1-0] and the byte enables for the initial data phase. Byte enables are asserted when 0.

Table 9. Byte Enables and AD[1-0] Encodings

AD[1-0]	Starting Byte	Valid BE[3-0] Combinations
00	Byte 0	xxx0 or 1111
01	Byte 1	xx01 or 1111
10	Byte 2	x011 or 1111
11	Byte 3	0111 or 1111

4.5.1 I/O Write Transactions

The DSP can request for the PCI to perform an I/O write operation by doing the following:

1. Reading the master configuration/IO access command register (PCIMCFGCMD) and ensuring that the READY bit (bit 31) is asserted.
2. Writing the data for the configuration write to the master configuration/IO access data register (PCIMCFGDAT), through the DSP register interface.
3. Writing the address for the configuration write to the master configuration/IO access address register (PCIMCFGADR).
4. Writing to the master configuration/IO access command register (PCIMCFGCMD) with the TYPE field cleared to 0 (I/O transaction), the RD_WR field cleared to 0 (write), and the BYTE_EN fields set to the desired value.

When a request is made for I/O write operation, the PCI asserts PREQ on the PCI bus. Once PGNT is sampled asserted, the PCI asserts PFRAME and outputs the write address onto the AD pins and the I/O Write command (3h) onto the PCBE pins. On the next cycle, the PCI asserts PIRDY, de-asserts PFRAME, and outputs the data to be written. The PCI will never insert wait states during a transfer but will respond to wait states as controlled by PTRDY. As is required, the PCI continually checks the bus for exceptions (master abort, target abort, retry, disconnect, latency timeout, parity error, system error) while the transfer is ongoing. A ready signal is returned to the DSP through the READY bit in the master configuration/IO access command register (PCIMCFGCMD) when the transfer is complete.

4.5.2 I/O Read Transactions

The DSP can request for the PCI to perform an I/O read operation by doing the following:

1. Reading the master configuration/IO access command register (PCIMCFGCMD) and ensuring that the READY bit (bit 31) is asserted.
2. Writing the address for the configuration write to the master configuration/IO access address register (PCIMCFGADR).
3. Writing to the master configuration/IO access command register (PCIMCFGCMD) with the TYPE field cleared to 0 (configuration transaction), the RD_WR field set to 1 (read), and the BYTE_EN fields set to the desired value.

When a request is made for I/O read operation, the PCI asserts $\overline{\text{PREQ}}$ on the PCI bus. Once $\overline{\text{PGNT}}$ is sampled asserted, the PCI asserts $\overline{\text{PFRAME}}$ and outputs the read address onto the AD pins and the I/O read command (2h) onto the PCBE pins. On the next cycle, the PCI asserts $\overline{\text{PIRDY}}$, and de-asserts $\overline{\text{PFRAME}}$. The PCI will never insert wait states during a transfer but will respond to wait states as controlled by $\overline{\text{PTRDY}}$. As is required, the PCI continually checks the bus for exceptions (master abort, target abort, retry, disconnect, latency timeout, parity error, system error) while the transfer is ongoing. When the transfer is complete, the data is returned to the master configuration/IO access data register (PCIMCFGDAT) and the READY bit in the master configuration/IO access command register (PCIMCFGCMD) is set.

4.6 Master Memory Operations

In the memory access, bits AD[31-2] are used to address the PCI device, the function within the device and a DWORD in the function's memory space. Bits AD[1-0] are ignored. However, bits AD[1-0] indicate the order in which the PCI is requesting the data to be transferred.

4.6.1 Memory Write Transactions

The DSP can request for the PCI to perform a memory write operation by making a memory write access to PCI master memory map. When a request is made for a memory write operation, the PCI asserts $\overline{\text{PREQ}}$ on the PCI bus. Once $\overline{\text{PGNT}}$ is sampled asserted; the PCI asserts $\overline{\text{PFRAME}}$ and outputs the write address onto the AD pins and the memory write command (7h) onto the PCBE pins. On the next cycle, the PCI asserts $\overline{\text{PIRDY}}$ and outputs the first word of data to be written. If the transfer only consists of a single data phase, $\overline{\text{PFRAME}}$ is de-asserted as required by the PCI specification coincident with the assertion of $\overline{\text{PIRDY}}$. Otherwise, $\overline{\text{PFRAME}}$ continues to be asserted until the next to the last data phase completes. The PCI will never insert wait states during a burst but will respond to wait states as controlled by $\overline{\text{PTRDY}}$. As is required, the PCI continually checks the bus for exceptions (master abort, target abort, retry, disconnect, latency timeout, parity error, system error) while the transfer is ongoing. As the burst progresses, an internal ready signal is returned from the DSP master interface for each successful data phase completion until the entire burst is finished.

Section 4.3 provides an example that describes how to program the EDMA to perform a master memory write operation.

4.6.2 Memory Read Transactions

The DSP can request for the PCI to perform a memory read operation by making a memory read access to PCI master memory map. When a request is made for a memory read operation, $\overline{\text{PREQ}}$ is asserted and the read burst is performed following the same behavior as for the write burst (no wait states, etc.). During the address phase, the memory read (6h), memory read line (Eh), or memory read multiple command (Ch) is output on the PCBE pins depending on the length of the transfer and the cache line size. The byte enables that were registered from the DSP master interface are inverted and output during the data phases. Unlike the write burst, the byte enables on the PCI bus will not change as the transaction progresses. This will not cause problems since bursts are only performed to prefetchable regions of memory. As the burst progresses, an internal ready signal is returned from the DSP master interface along with the read data for each successful data phase completion until the entire burst is finished.

Section 4.3 provides an example that describes how to program the EDMA to perform a master memory read operation.

5 PCI Registers

There are three types of PCI registers:

- PCI Configuration Registers–directly programmable by an external PCI host, and indirectly programmable by the DSP through the PCI memory-mapped registers. See Section 5.1.
- PCI Memory-Mapped Registers–directly programmable by the DSP, and indirectly programmable by an external PCI host through base address registers. See Section 5.2.
- PCI Configuration Hook Registers-directly programmable by the DSP, not accessible by an external PCI host. See Section 5.3.

The following sections describe the various software-accessible registers that are contained within the PCI.

5.1 PCI Configuration Registers

The DSP supports all standard PCI configuration registers. These registers, which can be directly accessed by the external PCI host through Type 0 configuration read and write transactions, contain the standard PCI configuration information (vendor identification, device identification, class code, revision number, base addresses, etc.). The DSP can access these registers indirectly through the PCI memory-mapped registers.

Depending on the boot and device configuration settings at device reset, some of the PCI configuration registers can be auto-loaded from an I2C EEPROM at device reset or can be initialized with default values.

If I2C EEPROM auto-initialization is not used, the PCI configuration registers are initialized with their default values. If auto-initialization is used, the PCI configuration registers cannot be properly accessed by the host until they are fully read from the I2C EEPROM. PCI host access to the PCI configuration registers before the completion of auto-initialization results in a disconnect with retry. For more details, see Section 2.12.3.

Table 10. PCI Configuration Registers

Offset	Register[1] Byte 3	Byte 2	Byte 1	Byte 0	Section
0h	Device ID		Vendor ID		Section 5.1.2/Section 5.1.1
4h	PCI Status		PCI Command		Section 5.1.4/Section 5.1.3
8h	Class Code			Revision ID	Section 5.1.6/Section 5.1.5
Ch	Built-In Self-Test	Header Type	Latency Timer	Cache Line Size	Section 5.1.10/Section 5.1.9/ Section 5.1.8/Section 5.1.7
10h	Base Address 0 (8 Mbyte prefetchable)				Section 5.1.11
14h	Base Address 1 (4 Mbyte prefetchable)				Section 5.1.11
18h	Base Address 2 (4 Mbyte prefetchable)				Section 5.1.11
1Ch	Base Address 3 (8 Mbyte prefetchable)				Section 5.1.11
20h	Base Address 4 (8 Mbyte prefetchable)				Section 5.1.11
24h	Base Address 5 (8 Mbyte prefetchable)				Section 5.1.11
28h	Reserved				
2Ch	Subsystem ID		Subsystem Vendor ID		Section 5.1.12/Section 5.1.13
34h	Reserved			Capabilities Pointer	Section 5.1.14
3Ch	Maximum Latency	Minimum Grant	Interrupt Pin	Interrupt Line	Section 5.1.18/Section 5.1.17/ Section 5.1.16/Section 5.1.15
40h-FFh	Reserved				

(1) Shaded registers can be auto-loaded from an I2C EEPROM.

5.1.1 Vendor Identification Register

The vendor identification register is shown in Figure 12 and described in Table 11.

Figure 12. Vendor Identification Register

15	0
VEND_ID	
R-104Ch	

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

Table 11. Vendor Identification Register Field Descriptions

Bit	Field	Value	Description
15-0	VEND_ID	104Ch	Vendor ID bits. Uniquely identifies the manufacturer of the device.

5.1.2 Device Identification Register

The device identification register is shown in Figure 13 and described in Table 12.

Figure 13. Device Identification Register

15	0
DEV_ID	
R-B003h	

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

Table 12. Device Identification Register Field Descriptions

Bit	Field	Value	Description
15-0	DEV_ID	B003h	Device ID bits. Identifies a specific device from the manufacturer.

5.1.3 PCI Command Register

The PCI command register is shown in Figure 14 and described in Table 13.

Figure 14. PCI Command Register

15					11	10	9	8
Reserved						INT_DIS	FAST_BTOB_EN	SERR_EN
R-0						R/W-0	R-0	R/W-0
7	6	5	4	3	2	1	0	
WAITCYCLECNTL	PAR_ERR_RES	VGA_PAL_SNP	MEM_WRINV_EN	SP_CYCL	BUS_MS	MEM_SP	IO_SP	
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	

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

Table 13. PCI Command Register Field Descriptions

Bit	Field	Value	Description
15-11	Reserved	0	Reserved
10	INT_DIS		$\overline{\text{PINTA}}$ disable bit. Controls whether or not the device can assert the $\overline{\text{PINTA}}$ pin. This bit is a disable for the output driver on the $\overline{\text{PINTA}}$ pin.
		0	Interrupt condition appears on the external $\overline{\text{PINTA}}$ pin.
		1	Interrupt condition does not appear on the external $\overline{\text{PINTA}}$ pin.
9	FAST_BTOB_EN	0	Fast back-to-back enable bit. Controls whether or not the device is allowed to perform back-to-back writes to different targets. The PCI will not perform fast back-to-back transactions; therefore, this bit is hardwired to 0.
8	SERR_EN		$\overline{\text{PSERR}}$ enable bit. This bit is an enable for the output driver on the $\overline{\text{PSERR}}$ pin.
		0	If this bit is cleared and a system error condition is set inside the PCI, the error signal does not appear on the $\overline{\text{SERR}}$ pin.
		1	The error signal appears on the $\overline{\text{SERR}}$ pin.
7	WAITCYCLECNTL	0	Waite cycle control bit. Indicates whether or not the device performs address stepping. The PCI does not support address stepping; therefore this bit is hardwired to 0.
6	PAR_ERR_RES		Parity error response bit. Controls whether or not the device responds to detected parity errors.
		0	The PCI sets the detected parity error bit (DET_PAR_ERR) in the PCI status register when an error is detected, but does not assert $\overline{\text{PPERR}}$ and continues normal operation.
		1	The PCI responds normally to parity errors.
5	VGA_PAL_SNP	0	VGA palette snoop bit. This bit is not applicable for the PCI and is hardwired to 0.
4	MEM_WRINV_EN		Memory write and invalidate enable bit. This bit enables the device to use the Memory Write and Invalidate command.
		0	The PCI does not attempt to use the Memory Write and Invalidate command.
		1	The PCI uses the Memory Write and Invalidate command.
3	SP_CYCL		Special cycle bit. Controls the device's response to special cycle commands.
		0	The device ignores all special cycle commands.
		1	The device monitors special cycle commands.
2	BUS_MS		Bus master bit. This bit enables the device to act as a PCI bus master.
		0	The device does not act as a master on the PCI bus.
		1	The device acts as a master on the PCI bus.
1	MEM_SP		Memory access bit. This bit enables the device to respond to memory accesses within its address space.
		0	The PCI does not respond to memory-mapped accesses.
		1	The PCI responds to memory-mapped accesses.
0	IO_SP	0	IO access bit. This bit enables the device to respond to I/O accesses within its address space. The PCI does not support I/O accesses as a slave; therefore, this bit is hardwired to 0.

5.1.4 PCI Status Register

The PCI status register is shown in Figure 15 and described in Table 14is initialized by certain bits in the PCI command register.

Figure 15. PCI Status Register

15	14	13	12	11	10	9	8
DET_PAR_ERR	SIG_SYS_ERR	RCV_MS_ABRT	RCV_TGT_ABRT	SIG_TGT_ABRT	DEVSEL_TIM		MS_DPAR_ERR
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1		R/W-0
7	6	5	4	3	2		0
FAST_BTOB_CAP	Reserved	66MHZ_CAP	CAP_LIST_IMPL	INT_STAT	Reserved		
R-0	R-0	R-0	R-1	R-0	R-0		

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

Table 14. PCI Status Register Field Descriptions

Bit	Field	Value	Description
15	DET_PAR_ERR	0-1	Detected parity error bit. This bit is set by the PCI to indicate that it detected a parity error, which was not necessarily reported on $\overline{\text{PPERR}}$ if parity reporting is disabled.
14	SIG_SYS_ERR	0-1	Signaled system error ($\overline{\text{PSERR}}$) bit. This bit is set by the PCI to indicate that it signaled a system error on the $\overline{\text{PSERR}}$ pin.
13	RCV_MS_ABRT	0-1	Received master abort bit. This bit is set by the PCI master unit in the PCI to indicate that it terminated a transaction with a master abort.
12	RCV_TGT_ABRT	0-1	Received target abort bit. This bit is set by the PCI master unit in the PCI to indicate that it has received a target abort when acting as a bus master.
11	SIG_TGT_ABRT	0	Signaled target abort bit. This bit is always 0 because the PCI cannot issue a target abort.
10-9	DEVSEL_TIM	2h	Device select ($\overline{\text{PDEVSEL}}$) timing bits. These bits indicate the decode response time capability of the device. The PCI decode logic supports medium $\overline{\text{PDEVSEL}}$ timing; therefore, these bits are hardwired to 01.
8	MS_DPAR_ERR	0-1	Master data parity error bit. This bit is set by the PCI when all of the following conditions are met: 1. The PCI asserted $\overline{\text{PPERR}}$ or observed $\overline{\text{PPERR}}$ asserted. 2. The PCI was the bus master during the observed $\overline{\text{PPERR}}$ assertion. 3. The parity error response bit (PAR_ERR_RES) is set.
7	FAST_BTOB_CAP	0	Fast back-to-back capable bit. This bit indicates that the device is capable of performing fast back-to-back transactions. The PCI does not support fast back-to-back transactions; therefore, this bit is hardwired to 0.
6	Reserved	0	Reserved
5	66MHZ_CAP	0-1	66MHz capable bit. This bit indicates whether or not the interface is capable of meeting the 66-MHZ PCI timing requirements.
4	CAP_LIST_IMPL	0-1	Capabilities list implemented bit. This bit indicates whether or not the interface provides at least one capabilities list.
3	INT_STAT	0-1	Interrupt status bit. This bit indicates the current interrupt status for the function as generated by the interrupt registers in the back end interface. If this bit is set and INT_DIS in the PCI command register is cleared, the $\overline{\text{PINTA}}$ pin will be asserted low. INT_DIS has no effect on the value of this bit.
2-0	Reserved	0	Reserved

5.1.5 Revision Identification Register

The revision identification register is shown in Figure 16 and described in Table 15.

Figure 16. Revision Identification Register

7							0
Revision ID							
R-1							

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

Table 15. Revision Identification Register Field Descriptions

Bit	Field	Value	Description
7-0	Revision ID	0-FFh	Revision ID bits. Identifies revision of the device.

5.1.6 Class Code Register

The class code register is shown in Figure 17 and described in Table 16.

Figure 17. Class Code Register

23							16	15							8	7							0
BASE_CLASS								SUB_CLASS								REG_LVL							
R-11h								R-80h								R-0							

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

Table 16. Class Code Register Field Descriptions

Bit	Field	Value	Description
23-16	BASE_CLASS	0-FFh	Base class bits.
15-8	SUB_CLASS	0-FFh	Sub-class bits.
7-0	REG_LVL	0-FFh	Register-level programming interface bits.

5.1.7 Cache Line Size Register

The cache line size register is shown in Figure 18 and described in Table 17.

Figure 18. Cache Line Size Register

7							0
CACHELN_SIZ							
R/W-0							

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

Table 17. Cache Line Size Register Field Descriptions

Bit	Field	Value	Description
7-0	CACHELN_SIZ	0-FFh	Cache line size bits. This register is provided so that the host can inform the device of the cache line size in units of 32-bit words. The value of this register is used by the PCI as a master device to determine whether to use Memory Write, Memory Write and Invalidate, Read, Read Line, or Read Multiple commands for accessing memory. This register is also used by the slave state machine to determine the size of prefetches that are performed on the Slave Back End Interface. Supported values for this register are as listed. Writing an unsupported value to this register results in the value cleared to 0, as specified in the *PCI Local Bus Specification*.
		0	Disabled
		1h-3h	Unsupported value
		4h	Cache Line is 16 bytes
		5h-7h	Unsupported value
		8h	Cache Line is 32 bytes
		9h-Fh	Unsupported value
		10h	Cache Line is 64 bytes
		11h-1Fh	Unsupported value
		20h	Cache Line is 128 bytes
		21h-FFh	Unsupported value

5.1.8 Latency Timer Register

The latency timer register is shown in Figure 19 and described in Table 18.

Figure 19. Latency Timer Register

7							0
LAT_TMR							
R/W-0							

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

Table 18. Latency Timer Register Field Descriptions

Bit	Field	Value	Description
7-0	LAT_TMR	0-FFh	Latency timer bits. This register is provided so that the host can restrict the continued usage of the PCI bus by a master involved in a multiple data cycle transaction after its $\overline{\text{PGNT}}$ has been removed. The host is required to write a value into this register indicating the maximum number of PCI cycles that the master can hold the bus (beginning from the assertion of $\overline{\text{PFRAME}}$). If $\overline{\text{PGNT}}$ is never removed during the transaction, the value in the latency timer value will not be used. Since the PCI supports transactions with multiple data cycles, the latency timer register is implemented. The latency timer register is initialized with all zeroes at reset. This register is not cleared on software reset.

5.1.9 Header Type Register

The header type register is shown in Figure 20 and described in Table 19.

Figure 20. Header Type Register

7							0
HDR_TYPE							
R-0							

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

Table 19. Header Type Register Field Descriptions

Bit	Field	Value	Description
7-0	HDR_TYPE	0-FFh	Header type bits. Identifies the layout of bytes 10h through 3Fh, and if the device is single or multi-function.

5.1.10 Built-In Self-Test Register

The built-in self-test register is shown in Figure 21 and described in Table 20.

Figure 21. Built-In Self-Test Register

7							0
BIST							
R-0							

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

Table 20. Built-In Self-Test Register Field Descriptions

Bit	Field	Value	Description
7-0	BIST	0	Built-in self test bits. Hardwired to 0.

5.1.11 Base Address Registers

The base address registers are shown in Figure 22 through Figure 27 and described in Table 21.

Figure 22. Base Address 0 Register

31															16
ADDR															
R/W-0															

15												4	3	2	1	0
ADDR													PREFETCH	TYPE		IOMEM_SP_IND
R/W-0													R-1	R-0		R-0

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

Figure 23. Base Address 1 Register

31															16
ADDR															
R/W-0															

15												4	3	2	1	0
ADDR													PREFETCH	TYPE		IOMEM_SP_IND
R/W-0													R-0	R-0		R-0

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

Figure 24. Base Address 2 Register

31															16
ADDR															
R/W-0															

15												4	3	2	1	0
ADDR													PREFETCH	TYPE		IOMEM_SP_IND
R/W-0													R-0	R-0		R-0

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

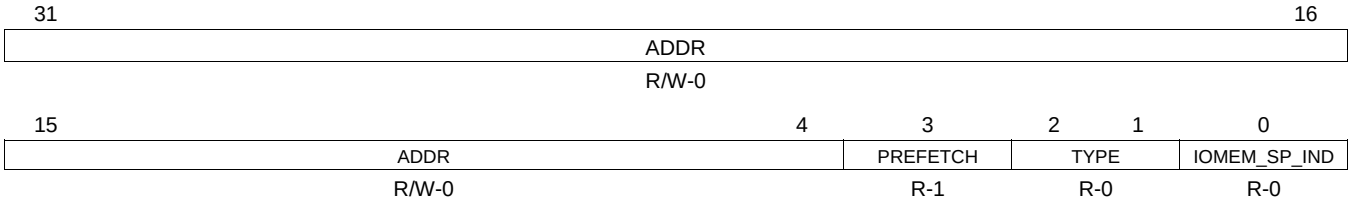
Figure 25. Base Address 3 Register

31															16
ADDR															
R/W-0															

15												4	3	2	1	0
ADDR													PREFETCH	TYPE		IOMEM_SP_IND
R/W-0													R-1	R-0		R-0

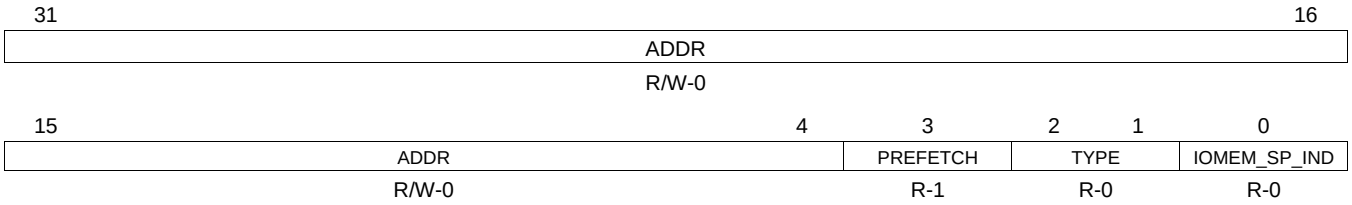
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 26. Base Address 4 Register



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

Figure 27. Base Address 5 Register



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

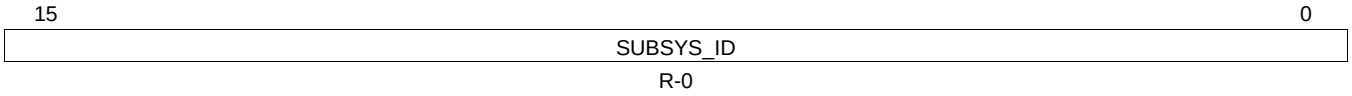
Table 21. Base Address *n* Registers Field Descriptions

Bit	Field	Value	Description
31-4	ADDR	0-FFF FFFFh	Address bits. These bits can be written by the host to allow initialization of the base address at startup. The writeability of individual bits is determined by the corresponding bit in the base address *n* mask register (PCIBAR*n*MSK).
3	PREFETCH	0-1	Prefetchable bit. Specifies whether or not the memory space controlled by this base address register is prefetchable. This bit reflects the value that is input from the PREFETCH_EN bit in the base address *n* mask register (PCIBAR*n*MSK).
2-1	TYPE	0	Type bits. Indicates the size of the base address register/decoder. This version of the PCI only supports 32-bit addressing, so these bits are hardwired to 0.
0	IOMEM_SP_IND	0	IO/Memory space indicator bit. Indicates whether the base address maps into the host's memory or I/O space. This version of the PCI only supports memory-mapped base address registers, so this bit is hardwired to 0.

5.1.12 Subsystem Identification Register

The subsystem identification register is shown in Figure 28 and described in Table 22.

Figure 28. Subsystem Identification Register



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

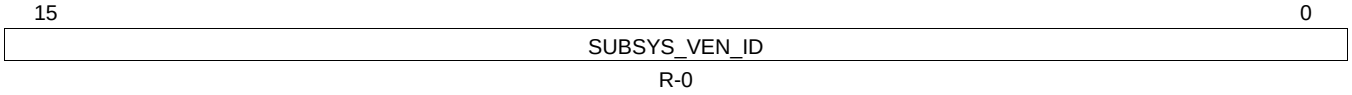
Table 22. Subsystem Identification Register Field Descriptions

Bit	Field	Value	Description
15-0	SUBSYS_ID	0	Subsystem ID bits. Identifies the board level device. The Subsystem ID is specified by the board level manufacturer.

5.1.13 Subsystem Vendor Identification Register

The subsystem vendor identification register is shown in Figure 29 and described in Table 23.

Figure 29. Subsystem Vendor Identification Register



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

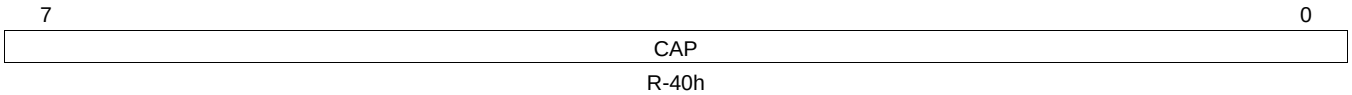
Table 23. Subsystem Vendor Identification Register Field Descriptions

Bit	Field	Value	Description
15-0	SUBSYS_VEN_ID	0	Subsystem vendor ID bits. Identifies the board level manufacturer. The Subsystem vendor ID is specified by the PCI Special Interest Group.

5.1.14 Capabilities Pointer Register

The capabilities pointer register is shown in Figure 30 and described in Table 24.

Figure 30. Capabilities Pointer Register



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

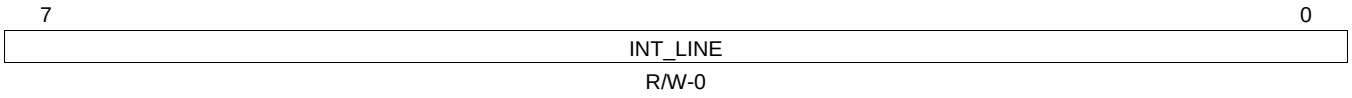
Table 24. Capabilities Pointer Register Field Descriptions

Bit	Field	Value	Description
7-0	CAP	0-FFh	Capabilities pointer bits. Specifies the address in configuration space where the first entry in the capabilities list is located.

5.1.15 Interrupt Line Register

The interrupt line register is shown in Figure 31 and described in Table 25.

Figure 31. Interrupt Line Register



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

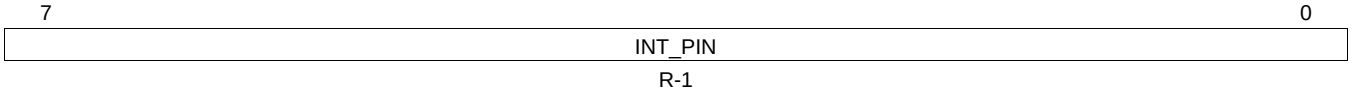
Table 25. Interrupt Line Register Field Descriptions

Bit	Field	Value	Description
7-0	INT_LINE	0-FFh	Interrupt line bits. This value is written by the host and indicates to which input of the system interrupt controller the PCI interrupt pin is connected.

5.1.16 Interrupt Pin Register

The interrupt pin register is shown in Figure 32 and described in Table 26.

Figure 32. Interrupt Pin Register



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

Table 26. Interrupt Pin Register Field Descriptions

Bit	Field	Value	Description
7-0	INT_PIN	1	Interrupt pin bits. Specifies the interrupt pin the device uses. This bit is hardwired to 01h in the PCI to indicate that interrupt A ($\overline{\text{PINTA}}$) is used.

5.1.17 Minimum Grant Register

The minimum grant register is shown in Figure 33 and described in Table 27.

Figure 33. Minimum Grant Register

7							0
MIN_GRNT							
R/W-0							

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

Table 27. Minimum Grant Register Field Descriptions

Bit	Field	Value	Description
7-0	MIN_GRNT	0-FFh	Minimum grant bits. Specifies the length of the burst period for the device in 0.25 µsec units.

5.1.18 Maximum Latency Register

The maximum latency register is shown in Figure 34 and described in Table 28.

Figure 34. Maximum Latency Register

7							0
MAX_LAT							
R/W-0							

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

Table 28. Maximum Latency Register Field Descriptions

Bit	Field	Value	Description
7-0	MAX_LAT	0-FFh	Maximum latency bits. Specifies how often the device needs to gain access to the PCI bus in 0.25 µsec units.

5.2 PCI Memory-Mapped Registers

The PCI memory-mapped registers listed in Table 29 provide the DSP access to control and status information within the PCI. The DSP can access all of the PCI configuration registers through the PCI memory-mapped registers. An external PCI host can indirectly access the PCI memory-mapped registers through base address registers.

Some of the memory-mapped registers are located inside the peripheral clock domain and the remaining registers are located inside the PCI clock (PCLK) domain. Registers in the PCI clock domain can only be accessed if both the PCI and peripheral clocks are running. Registers in the peripheral clock domain can be accessed even if the PCI clock is not running.

Table 29. PCI Memory-Mapped Registers

Offset	Acronym	Register Description	Section
010h	PCISTATSET(1)	Status Set Register	Section 5.2.1
014h	PCISTATCLR(1)	Status Clear Register	Section 5.2.1
020h	PCIHINTSET(1)	Host Interrupt Enable Set Register	Section 5.2.2
024h	PCIHINTCLR(1)	Host Interrupt Enable Clear Register	Section 5.2.2
030h	PCIDINTSET(1)	DSP Interrupt Enable Set Register	Section 5.2.3
034h	PCIDINTCLR(1)	DSP Interrupt Enable Clear Register	Section 5.2.3
100h	PCIVENDEVMIR	Vendor ID/Device ID Mirror Register	Section 5.2.4
104h	PCICSRMIR	Command/Status Mirror Register	Section 5.2.5
108h	PCICLREVMIR	Class Code/Revision ID Mirror Register	Section 5.2.6
10Ch	PCICLINEMIR	BIST/Header Type/Latency Timer/Cacheline Size Mirror Register	Section 5.2.7
110h	PCIBAR0MSK	Base Address 0 Mask Register	Section 5.2.8
114h	PCIBAR1MSK	Base Address 1 Mask Register	Section 5.2.8
118h	PCIBAR2MSK	Base Address 2 Mask Register	Section 5.2.8
11Ch	PCIBAR3MSK	Base Address 3 Mask Register	Section 5.2.8
120h	PCIBAR4MSK	Base Address 4 Mask Register	Section 5.2.8
124h	PCIBAR5MSK	Base Address 5 Mask Register	Section 5.2.8
12Ch	PCISUBIDMIR	Subsystem Vendor ID/Subsystem ID Mirror Register	Section 5.2.9
134h	PCICPBPTRMIR	Capabilities Pointer Mirror Register	Section 5.2.10
13Bh	PCILGINTMIR	Maximum Latency/Minimum Grant/Interrupt Pin/Interrupt Line Mirror Register	Section 5.2.11
180h	PCISLVCNTL	Slave Control Register	Section 5.2.12
1C0h	PCIBAR0TRL	Slave Base Address 0 Translation Register	Section 5.2.13
1C4h	PCIBAR1TRL	Slave Base Address 1 Translation Register	Section 5.2.13
1C8h	PCIBAR2TRL	Slave Base Address 2 Translation Register	Section 5.2.13
1CCh	PCIBAR3TRL	Slave Base Address 3 Translation Register	Section 5.2.13
1D0h	PCIBAR4TRL	Slave Base Address 4 Translation Register	Section 5.2.13
1D4h	PCIBAR5TRL	Slave Base Address 5 Translation Register	Section 5.2.13
1E0h	PCIBAR0MIR	Base Address 0 Mirror Register	Section 5.2.14
1E4h	PCIBAR1MIR	Base Address 1 Mirror Register	Section 5.2.14
1E8h	PCIBAR2MIR	Base Address 2 Mirror Register	Section 5.2.14
1ECh	PCIBAR3MIR	Base Address 3 Mirror Register	Section 5.2.14
1F0h	PCIBAR4MIR	Base Address 4 Mirror Register	Section 5.2.14
1F4h	PCIBAR5MIR	Base Address 5 Mirror Register	Section 5.2.14
300h	PCIMCFGDAT	Master Configuration/IO Access Data Register	Section 5.2.15.1
304h	PCIMCFGADR	Master Configuration/IO Access Address Register	Section 5.2.15.2

(1) These PCI memory-mapped registers are located in the peripheral clock domain. Registers in the peripheral clock domain can be accessed even if the PCI clock ($\overline{\text{PCLK}}$) is not running. Other memory-mapped registers are located in the PCI clock domain, and can only be accessed if both the PCI and the peripheral clocks are running.

Table 29. PCI Memory-Mapped Registers (continued)

Offset	Acronym	Register Description	Section
308h	PCIMCFGCMD	Master Configuration/IO Access Command Register	Section 5.2.15.3
310h	PCIMSTCFG	Master Configuration Register	Section 5.2.16
314h-390h	PCIADDSUB[0-31](1)	PCI Address Substitution Registers	Section 5.2.17

5.2.1 Status Set and Status Clear Registers (PCISTATSET/PCISTATCLR)

The PCI includes an internal status register that is not directly writable by the DSP for software simplification. As a result, two registers, the status set register (PCISTATSET) and the status clear register (PCISTATCLR), are provided to set or clear bits in the internal status register.

PCISTATSET and PCISTATCLR both return the contents of the internal status register during reads. Writing a 1 to any of the bits in PCISTATSET sets the corresponding bit in the internal status register. Writing a 1 to any of the bits in PCISTATCLR clears the corresponding bit in the internal status register as long as the corresponding condition that sets that bit is not also asserted.

Bits in the status register cannot be cleared unless the underlying condition that caused the bit to be set has also been cleared. This is important for the $\overline{\text{PINTA}}$, $\overline{\text{PPERR}}$, and $\overline{\text{PSERR}}$ interrupts that are generated and accessed on a level-sensitive external pin. Additionally, there is a synchronization delay of 3 peripheral clocks present between the time that a level-sensitive status condition is cleared in the PCI clock domain and when that condition will be cleared in the peripheral clock domain. Interrupt service routines need to be designed to include this delay.

The PCISTATSET and PCISTATCLR is shown in Figure 35 and described during reads in Table 30. The PCISTATSET during writes is described in Table 31; the PCISTATCLR during writes is described in Table 32.

Figure 35. Status Set and Status Clear Registers (PCISTATSET/PCISTATCLR)

31	28	27	26	25	24
Reserved		SOFT_INT3	SOFT_INT2	SOFT_INT1	SOFT_INT0
R-0		R/W-0	R/W-0	R/W-0	R/W-0

23	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved	PERR_DET	SERR_DET	Reserved		MS_ABRT_DET	TGT_ABRT_DET	Reserved
R-0	R/W-0	R/W-0	R-0		R/W-0	R/W-0	R-0

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

Table 30. Status Set and Status Clear Registers (PCISTATSET/PCISTATCLR) Field Descriptions During Reads

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-24	SOFT_INT*n*		Software interrupt status bit. Reading these bits returns the value of the SOFT_INT*n* bits in the internal status register.
		0	Status bit is not set.
		1	Status bit is set.
23-7	Reserved	0	Reserved
6	PERR_DET		Parity error detect status bit. Reading this bit returns the value of the PERR_DET bit in the internal status register.
		0	Status bit is not set.
		1	Status bit is set.
5	SERR_DET		System error detect status bit. Reading this bit returns the value of the SERR_DET bit in the internal status register.
		0	Status bit is not set.
		1	Status bit is set.
4-3	Reserved	0	Reserved
2	MS_ABRT_DET		Master abort detect status bit. Reading this bit returns the value of the MS_ABRT_DET bit in the internal status register.
		0	Status bit is not set.
		1	Status bit is set.
1	TGT_ABRT_DET		Target abort detect status bit. Reading this bit returns the value of the TGT_ABRT_DET bit in the internal status register.
		0	Status bit is not set.
		1	Status bit is set.
0	Reserved	0	Reserved

Table 31. Status Set Register (PCISTATSET) Field Descriptions During Writes

Bit	Field	Value	Description
31	Reserved	0	Reserved. Always write 0 to this bit.
30-28	Reserved	0	Reserved
27-24	SOFT_INT*n*		Software interrupt bits. Writing a 1 to these bits sets the SOFT_INT*n* bits in the internal status register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Set the SOFT_INT*n* bit in the internal status register.
23-7	Reserved	0	Reserved
6	PERR_DET		Parity error detect bit. Writing 1 to this bit sets the PERR_DET bit in the internal status register.
		0	A write of 0 has no effect.
		1	Set the PERR_DET bit in the internal status register.
5	SERR_DET		System error detect bit. Writing 1 to this bit sets the SERR_DET bit in the internal status register.
		0	A write of 0 has no effect.
		1	Set the SERR_DET bit in the internal status register.
4-3	Reserved	0	Reserved
2	MS_ABRT_DET		Master abort detect bit. Writing 1 to this bit sets the MS_ABRT_DET bit in the internal status register.
		0	A write of 0 has no effect.
		1	Set the MS_ABRT_DET bit in the internal status register.
1	TGT_ABRT_DET		Target abort detect bit. Writing 1 to this bit sets the TGT_ABRT_DET bit in the internal status register.
		0	A write of 0 has no effect.
		1	Set the TGT_ABRT_DET in the internal status register.
0	Reserved	0	Reserved

Table 32. Status Clear Register (PCISTATCLR) Field Descriptions During Writes

Bit	Field	Value	Description
31	Reserved	0	Reserved. Always write 0 to this bit.
30-28	Reserved	0	Reserved
27-24	SOFT_INT*n*		Software interrupt bits. Writing a 1 to these bits clears the SOFT_INT*n* bits in the internal status register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Clear the SOFT_INT*n* bit in the internal status register.
23-7	Reserved	0	Reserved
6	PERR_DET		Parity error detect bit. Writing 1 to this bit clears the PERR_DET bit in the internal status register.
		0	A write of 0 has no effect.
		1	Clear the PERR_DET bit in the internal status register.
5	SERR_DET		System error detect bit. Writing 1 to this bit clears the SERR_DET bit in the internal status register.
		0	A write of 0 has no effect.
		1	Clear the SERR_DET bit in the internal status register.
4-3	Reserved	0	Reserved
2	MS_ABRT_DET		Master abort detect bit. Writing 1 to this bit clears the MS_ABRT_DET bit in the internal status register.
		0	A write of 0 has no effect.
		1	Clears the MS_ABRT_DET bit in the internal status register.
1	TGT_ABRT_DET		Target abort detect bit. Writing 1 to this bit clears the TGT_ABRT_DET bit in the internal status register.
		0	A write of 0 has no effect.
		1	Clears the TGT_ABRT_DET in the internal status register.
0	Reserved	0	Reserved

5.2.2 Host Interrupt Enable Set and Enable Clear Registers (PCIHINTSET/PCIHINTCLR)

The PCI includes an internal host interrupt enable register that for software simplification is not directly writable by the host. As a result, two registers, the host interrupt enable set register (PCIHINTSET) and the host interrupt enable clear register (PCIHINTCLR), are provided to set or clear bits in the internal host interrupt enable register.

The host uses the internal host interrupt enable register to configure on which status conditions an interrupt will be generated to the host. A level-sensitive active-low interrupt is generated to the host on the $\overline{\text{PINTA}}$ pin if a bit in the internal status register, described in Section 5.2.1, is asserted and the corresponding bit in the internal host interrupt enable register is also asserted, as long as the PCI is in the D0 power state. Interrupt generation on the $\overline{\text{PINTA}}$ pin is disabled whenever the PCI is in the D1, D2, or D3 power states.

Writing a 1 to any of the bits in PCIHINTSET sets the corresponding bit in the internal host interrupt enable register. Writing a 1 to any of the bits in PCIHINTCLR clears the corresponding bit in the internal host interrupt enable register.

Reading PCIHINTSET returns the internal host interrupt enable register contents. Reading from PCIHINTCLR returns the masked host status that is the bitwise ANDing of the internal status register and the internal host interrupt enable register. PCIHINTCLR is typically read by the host to determine the interrupt source when the $\overline{\text{PINTA}}$ pin is asserted. An external PCI host can indirectly access the PCI memory-mapped registers through base address registers.

The PCIHINTSET and PCIHINTCLR is shown in Figure 36. The PCIHINTSET field descriptions are in Table 33; the PCIHINTCLR field descriptions are described in Table 34.

Figure 36. Host Interrupt Enable Set and Clear Registers (PCIHINTSET/PCIHINTCLR)

31	28	27	26	25	24
Reserved		SOFT_INT3	SOFT_INT2	SOFT_INT1	SOFT_INT0
R-0		R/W-0	R/W-0	R/W-0	R/W-0

23	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved	PERR_DET	SERR_DET	Reserved		MS_ABRT_DET	TGT_ABRT_DET	Reserved
R-0	R/W-0	R/W-0	R-0		R/W-0	R/W-0	R-0

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

Table 33. Host Interrupt Enable Set Register (PCIHINTSET) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-24	SOFT_INT*n*		Software interrupt bits. Writing a 1 to these bits sets the SOFT_INT*n* bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enable software interrupt SOFT_INT*n*.
23-7	Reserved	0	Reserved
6	PERR_DET		Parity error detect enable bit. Writing a 1 to this bit sets the PERR_EN bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enable the parity error detect interrupt.
5	SERR_DET		System error detect enable bit. Writing a 1 to this bit sets the SERR_DET bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enable the system error detect interrupt.
4-3	Reserved	0	Reserved
2	MS_ABRT_DET		Master abort detect enable bit. Writing a 1 to this bit sets the MS_ABRT_DET bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enable the master abort detect interrupt.
1	TGT_ABRT_DET		Target abort detect enable bit. Writing a 1 to this bit sets the TGT_ABRT_DET bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enable the target abort detect interrupt.
0	Reserved	0	Reserved

Table 34. Host Interrupt Enable Clear Register (PCIHINTCLR) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-24	SOFT_INT*n*		Software interrupt bits. Writing a 1 to these bits clears the SOFT_INT*n* bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disable software interrupt SOFT_INT*n*.
23-7	Reserved	0	Reserved
6	PERR_DET		Parity error detect enable bit. Writing a 1 to this bit clears the PERR_EN bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disable the parity error detect interrupt.
5	SERR_DET		System error detect enable bit. Writing a 1 to this bit clears the SERR_DET bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disable the system error detect interrupt.
4-3	Reserved	0	Reserved
2	MS_ABRT_DET		Master abort detect enable bit. Writing a 1 to this bit clears the MS_ABRT_DET bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disable the master abort detect interrupt.
1	TGT_ABRT_DET		Target abort detect enable bit. Writing a 1 to this bit clears the TGT_ABRT_DET bit in the internal host interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disable the target abort detect interrupt.
0	Reserved	0	Reserved

5.2.3 DSP Interrupt Enable Set and Enable Clear Registers (PCIDINTSET/PCIDINTCLR)

The PCI includes a DSP interrupt enable register that is not directly writable by the DSP for software simplification. As a result, two registers, the DSP interrupt enable set register (PCIDINTSET) and the DSP interrupt enable clear register (PCIDINTCLR) are provided to set or clear bits in the internal DSP interrupt enable register.

The DSP uses the internal DSP interrupt enable register to configure on which status conditions an interrupt will be generated to the DSP. An interrupt request is generated to the DSP via the PCIINT interrupt line if a bit in the internal status register, described in Section 5.2.1, is asserted and the corresponding bit in the DSP interrupt enable register is also asserted.

Writing a 1 to any of the bits in PCIDINTSET sets the corresponding bit in the internal DSP interrupt enable register. Writing a 1 to any of the bits in PCIDINTCLR clears the corresponding bit in the internal DSP interrupt enable register.

Reading PCIDINTSET returns the internal DSP interrupt enable register contents. Reading from PCIDINTCLR returns the masked DSP status that is the bitwise ANDing of the internal status register and the internal DSP interrupt enable register. PCIDINTCLR is typically read by the DSP to determine the source of a PCI interrupt.

The PCIDINTSET and PCIDINTCLR is shown in Figure 37. The PCIDINTSET field descriptions are described in Table 35; the PCIDINTCLR field descriptions are described in Table 36.

Figure 37. DSP Interrupt Enable Set and Clear Registers (PCIDINTSET/PCIDINTCLR)

31	28	27	26	25	24
Reserved		SOFT_INT3	SOFT_INT2	SOFT_INT1	SOFT_INT0
R-0		R/W-0	R/W-0	R/W-0	R/W-0

23	16
Reserved	
R-0	

15	8
Reserved	
R-0	

7	6	5	4	3	2	1	0
Reserved	PERR_DET	SERR_DET	Reserved		MS_ABRT_DET	TGT_ABRT_DET	Reserved
R-0	R/W-0	R/W-0	R-0		R/W-0	R/W-0	R-0

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

Table 35. DSP Interrupt Enable Set Register (PCIDINTSET) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-24	SOFT_INT*n*		Software interrupt bits. Writing a 1 to these bits sets the SOFT_INT*n* bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enable software interrupt SOFT_INT*n*.
23-7	Reserved	0	Reserved
6	PERR_DET		Parity error detect enable bit. Writing a 1 to this bit sets the PERR_EN bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enables the parity error detect interrupt.
5	SERR_DET		System error detect enable bit. Writing a 1 to this bit sets the SERR_DET bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enables the system error detect interrupt.
4-3	Reserved	0	Reserved
2	MS_ABRT_DET		Master abort detect enable bit. Writing a 1 to this bit sets the MS_ABRT_DET bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enables the master abort detect interrupt.
1	TGT_ABRT_DET		Target abort detect enable bit. Writing a 1 to this bit sets the TGT_ABRT_DET bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Enables the target abort detect interrupt.
0	Reserved	0	Reserved

Table 36. DSP Interrupt Enable Clear Register (PCIDINTCLR) Field Descriptions

Bit	Field	Value	Description
31-28	Reserved	0	Reserved
27-24	SOFT_INT*n*		Software interrupt bits. Writing a 1 to these bits clears the SOFT_INT*n* bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disable software interrupt SOFT_INT*n*.
23-7	Reserved	0	Reserved
6	PERR_DET		Parity error detect enable bit. Writing a 1 to this bit clears the PERR_EN bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disables the parity error detect interrupt.
5	SERR_DET		System error detect enable bit. Writing a 1 to this bit clears the SERR_DET bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disables the system error detect interrupt.
4-3	Reserved	0	Reserved
2	MS_ABRT_DET		Master abort detect enable bit. Writing a 1 to this bit clears the MS_ABRT_DET bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disables the master abort detect interrupt.
1	TGT_ABRT_DET		Target abort detect enable bit. Writing a 1 to this bit clears the TGT_ABRT_DET bit in the internal DSP interrupt enable register. Writing a 0 has no effect.
		0	A write of 0 has no effect.
		1	Disables the target abort detect interrupt.
0	Reserved	0	Reserved

5.2.4 Vendor Identification/Device Identification Mirror Register (PCIVENDEVMIR)

The vendor identification/device identification mirror register (PCIVENDEVMIR) is shown in Figure 38 and described in Table 37.

Figure 38. Vendor ID/Device ID Mirror Register (PCIVENDEVMIR)

31	16
DEV_ID	
R/W-B003h	

15	0
VEN_ID	
R/W-104Ch	

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

Table 37. Vendor ID/Device ID Mirror Register (PCIVENDEVMIR) Field Descriptions

Bit	Field	Value	Description
31-16	DEV_ID	0-FFFFh	Device ID bits. Identifies a specific device from the manufacturer.
15-0	VEN_ID	0-FFFFh	Vendor ID bits. Uniquely identifies the manufacturer of the device.

5.2.5 Command/Status Mirror Register (PCICSRMIR)

The command/status mirror register (PCICSRMIR) is shown in Figure 39 and described in Table 38.

Figure 39. Command/Status Mirror Register (PCICSRMIR)

31	30	29	28	27	26	25	24
DET_PAR_ERR	SIG_SYS_ERR	RCV_MS_ABRT	RCV_TGT_ABRT	SIG_TGT_ABRT	DEVSEL_TIM		MS_DPAR_ERR
R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R/W1C-0	R-1		R/W1C-0

23	22	21	20	19	18	16
FAST_BTOB_CAP	Reserved	66MHZ_CAP	CAP_LIST_IMPL	INT_STAT	Reserved	
R-0	R-0	R-0	R-1	R-0	R-0	

15	11	10	9	8
Reserved		INT_DIS	FAST_BTOB_EN	SERR_EN
R-0		R/W-0	R-0	R/W-0

7	6	5	4	3	2	1	0
WAITCYCLECNTL	PAR_ERR_RES	VGA_PAL_SNP	MEM_WRINV_EN	SP_CYCL	BUS_MS	MEM_SP	IO_SP
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

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

Table 38. Command/Status Mirror Register (PCICSRMIR) Field Descriptions

Bit	Field	Value	Description
31	DET_PAR_ERR	0-1	Detected parity error bit. This bit is set by the PCI to indicate that it detected a parity error, which was not necessarily reported on $\overline{\text{PPERR}}$ if parity reporting is disabled.
30	SIG_SYS_ERR	0-1	Signaled system error ($\overline{\text{PSERR}}$) bit. This bit is set by the PCI to indicate that it signaled a system error on the $\overline{\text{PSERR}}$ pin.
29	RCV_MS_ABRT	0-1	Received master abort bit. This bit is set by the PCI master unit in the PCI to indicate that it terminated a transaction with a master abort.
28	RCV_TGT_ABRT	0-1	Received target abort bit. This bit is set by the PCI master unit in the PCI to indicate that it has received a target abort when acting as a bus master.
27	SIG_TGT_ABRT	0-1	Signaled target abort bit. This bit is always 0 because the PCI cannot issue a target abort.
26-25	DEVSEL_TIM	2h	Device select ($\overline{\text{PDEVSEL}}$) timing bits. These bits indicate the decode response time capability of the device. The PCI decode logic supports medium $\overline{\text{PDEVSEL}}$ timing; therefore, these bits are hardwired to 01.
24	MS_DPAR_ERR	0-1	Master data parity error bit. This bit is set by the PCI when all of the following conditions are met: 1. The PCI asserted $\overline{\text{PPERR}}$ or observed $\overline{\text{PPERR}}$ asserted. 2. The PCI was the bus master during the observed $\overline{\text{PPERR}}$ assertion. 3. The parity error response bit (PAR_ERR_RES) is set.
23	FAST_BTOB_CAP	0	Fast back-to-back capable bit. This bit indicates that the device is capable of performing fast back-to-back transactions. The PCI does not support fast back-to-back transactions; therefore, this bit is hardwired to 0.
22	Reserved	0	Reserved
21	66MHZ_CAP	0	66MHz capable bit. This bit indicates whether or not the interface is capable of meeting the 66MHz PCI timing requirements.
20	CAP_LIST_IMPL	0	Capabilities list implemented bit. This bit indicates whether or not the interface provides at least one capabilities list.
19	INT_STAT	0-1	Interrupt status bit. This bit indicates the current interrupt status for the function as generated by the interrupt registers in the back end interface. If this bit is set and INT_DIS is cleared, the $\overline{\text{PINTA}}$ pin will be asserted low. INT_DIS has no effect on the value of this bit.
18-11	Reserved	0	Reserved. Always return 0.
10	INT_DIS	0-1	$\overline{\text{PINTA}}$ disable bit. This bit controls whether or not the device can assert the $\overline{\text{PINTA}}$ pin. This bit is a disable for the output driver on the $\overline{\text{PINTA}}$ pin. If this bit is set, the interrupt condition will not appear on the external $\overline{\text{PINTA}}$ pin.

Table 38. Command/Status Mirror Register (PCICSRMIR) Field Descriptions (continued)

Bit	Field	Value	Description
9	FAST_BTOB_EN	0	Fast back-to-back enable bit. This bit controls whether or not the device is allowed to perform back-to-back writes to different targets. The PCI will not perform fast back-to-back transactions; therefore, this bit is hardwired to a 0.
8	SERR_EN	0-1	$\overline{\text{PSERR}}$ enable bit. This bit is an enable for the output driver on the $\overline{\text{PSERR}}$ pin. If this bit is cleared, and a system error condition is set inside the PCI, the error signal will not appear on the $\overline{\text{SERR}}$ pin.
7	WAITCYCLECNTL	0	Waite cycle control bit. This bit indicates whether or not the device performs address stepping. The PCI does not support address stepping; therefore this bit is hardwired to 0.
6	PAR_ERR_RES	0-1	Parity error response bit. This bit controls whether or not the device responds to detected parity errors. If this bit is set, the PCI will respond normally to parity errors. If this bit is cleared, the PCI will set its Detected Parity Error bit (DET_PAR_ERR) when an error is detected, but does not assert $\overline{\text{PPERR}}$ and continues normal operation.
5	VGA_PAL_SNP	0	VGA Palette Snoop bit. This bit is not applicable for the PCI and is hardwired to a 0.
4	MEM_WRINV_EN	0-1	Memory write and invalidate enable bit. This bit enables the device to use the Memory Write and Invalidate command. If this bit is cleared, the PCI will not attempt to use the Memory Write and Invalidate command.
3	SP_CYCL	0	Special cycle bit. This bit controls the device's response to special cycle commands. If this bit is cleared, the device will ignore all special cycle commands. If this bit is set to 1, the device can monitor special cycle commands.
2	BUS_MS	0-1	Bus master bit. This bit enables the device to act as a PCI bus master. If this bit is cleared, the device will not act as a master on the PCI bus.
1	MEM_SP	0-1	Memory access bit. This bit enables the device to respond to memory accesses within its address space. If this bit is cleared, the PCI will not respond to memory mapped accesses.
0	IO_SP	0	IO access bit. This bit enables the device to respond to I/O accesses within its address space. The PCI does not support IO accesses as a slave; therefore, this bit is hardwired to 0.

5.2.6 Class Code/Revision Identification Mirror Register (PCICLREVMIR)

The class code/revision identification mirror register (PCICLREVMIR) is shown in Figure 40 and described in Table 39.

Figure 40. Class Code/Revision Identification Mirror Register (PCICLREVMIR)

31	24	23	16
BASE_CLASS		SUB_CLASS	
R/W-11h		R/W-80h	

15	8	7	0
REG_LVL		REV_ID	
R/W-0		R/W-1	

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

Table 39. Class Code/Revision Identification Mirror Register (PCICLREVMIR) Field Descriptions

Bit	Field	Value	Description
31-24	BASE_CLASS	0-FFh	Base class bits.
23-16	SUB_CLASS	0-FFh	Sub-class bits.
15-8	REG_LVL	0-FFh	Register-level programming interface bits.
7-0	REV_ID	0-FFh	Revision ID bits. Identifies a revision of the specific device.

5.2.7 BIST/Header Type/Latency Timer/Cacheline Size Mirror Register (PCICLINEMIR)

The BIST/header type/latency timer/cacheline size mirror register (PCICLINEMIR) is shown in Figure 41 and described in Table 40.

Figure 41. BIST/Header Type/Latency Timer/Cacheline Size Mirror Register (PCICLINEMIR)

31	24	23	16
BIST		HDR_TYPE	
R-0		R-0	

15	8	7	0
LAT_TMR		CACHELN_SIZ	
R/W-0		R/W-0	

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

Table 40. BIST/Header Type/Latency Timer/Cacheline Size Mirror Register (PCICLINEMIR) Field Descriptions

Bit	Field	Value	Description
31-24	BIST	0	Built-in self test bits. Hardwired to 0.
23-16	HDR_TYPE	0-FFh	Header type bits. Identifies the layout of bytes 10h through 3Fh, and if the device is single or multi-function.
15-8	LAT_TMR	0-FFh	Latency timer bits. The latency timer register is provided so that the host can restrict the continued usage of the PCI bus by a master involved in a multiple data cycle transaction after its $\overline{\text{PGNT}}$ has been removed. The host is required to write a value into this register indicating the maximum number of PCI cycles for which the master can hold the bus (beginning from the assertion of $\overline{\text{PFRAME}}$). If $\overline{\text{PGNT}}$ is never removed during the transaction, the value in the latency timer value will not be used. Since the PCI will support transactions with multiple data cycles, the latency timer register is implemented. The latency timer register is initialized with all zeroes at reset. This register is not cleared on software reset.
7-0	CACHELN_SIZ	0-FFh	Cache line size bits. This register is provided so that the host can inform the device of the cache line size in units of 32 bit words. The value of this register is used by the PCI as a master device to determine whether to use Memory Write, Memory Write and Invalidate, Read, Read Line, or Read Multiple commands for accessing memory. This register is also used by the slave state machine to determine the size of prefetches that are performed on the Slave Back End Interface. Supported values for this register are as listed. Writing an unsupported value to this register will result in the value cleared to 0, as specified in the *PCI Local Bus Specification*.
		0h	Disabled
		1h-3h	Unsupported value
		4h	Cache Line is 16 bytes
		5h-7h	Unsupported value
		8h	Cache Line is 32 bytes
		9h-Fh	Unsupported value
		10h	Cache Line is 64 bytes
		11h-1Fh	Unsupported value
		20h	Cache Line is 128 bytes
		21h-FFh	Unsupported value

5.2.8 Base Address Mask Registers (PCIBAR0MSK-PCIBAR5MSK)

The base address mask registers (PCIBAR0MSK-PCIBAR5MSK) control the size and prefetchability of the PCI slave windows. The base address mask registers are shown in Figure 42 through Figure 47 and described in Table 41.

Figure 42. Base Address 0 Mask Register (PCIBAR0MSK)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRMASK															
R/W-FF80h															

15	14	13	12	11	10	9	8
ADDRMASK							
R/W-00h							

7	6	5	4	3	2	1	0
ADDRMASK				PREFETCH_EN	Reserved		
R/W-0h				R-1	R-0		

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

Figure 43. Base Address 1 Mask Register (PCIBAR1MSK)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRMASK															
R/W-FFC0h															

15	14	13	12	11	10	9	8
ADDRMASK							
R/W-00h							

7	6	5	4	3	2	1	0
ADDRMASK				PREFETCH_EN	Reserved		
R/W-0h				R-0	R-0		

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

Figure 44. Base Address 2Mask Register (PCIBAR2MSK)

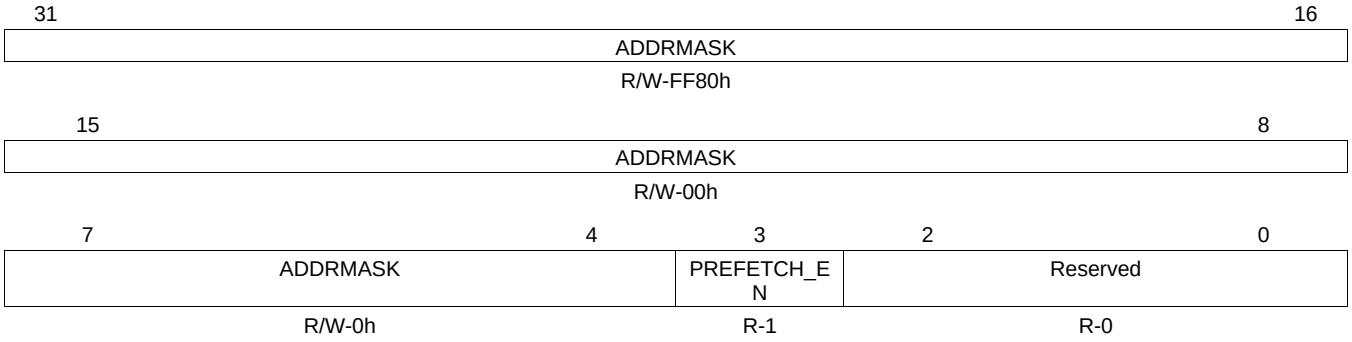
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
ADDRMASK															
R/W-FFC0h															

15	14	13	12	11	10	9	8
ADDRMASK							
R/W-00h							

7	6	5	4	3	2	1	0
ADDRMASK				PREFETCH_EN	Reserved		
R/W-0h				R-0	R-0		

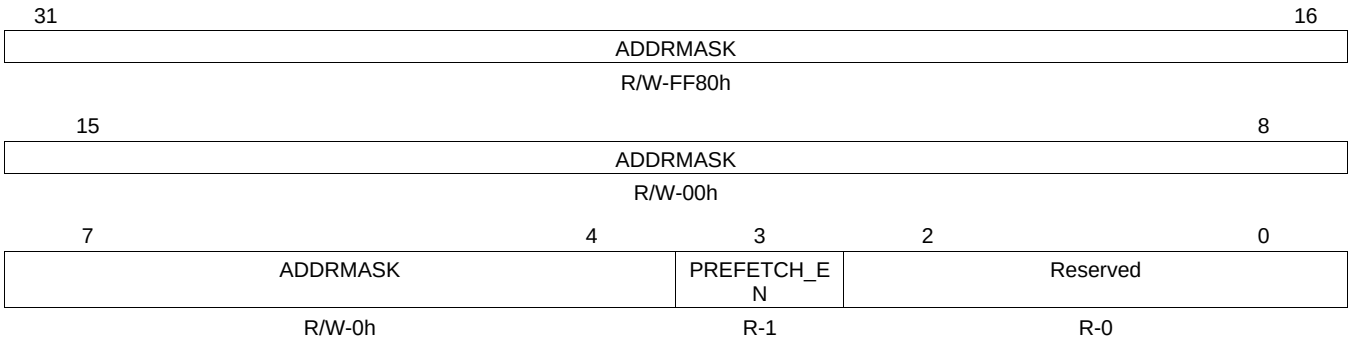
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 45. Base Address 3 Mask Register (PCIBAR3MSK)



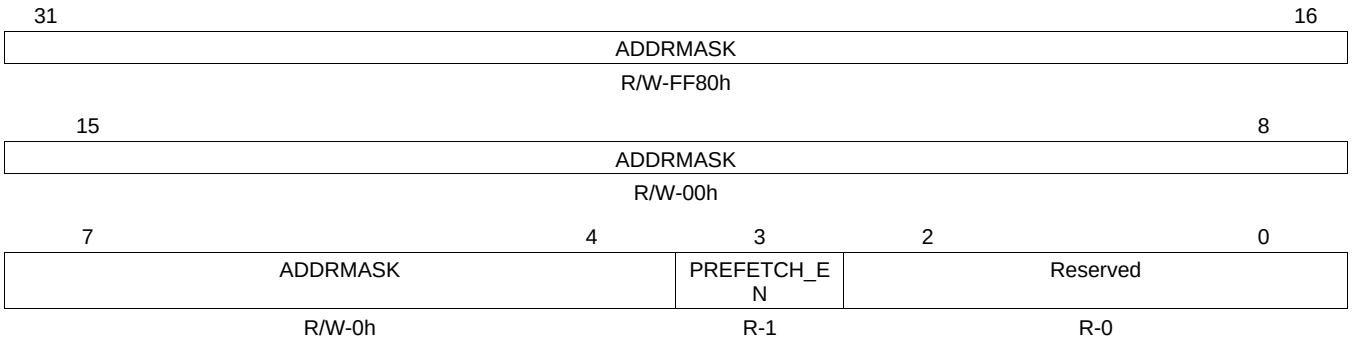
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 46. Base Address 4 Mask Register (PCIBAR4MSK)



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

Figure 47. Base Address 5 Mask Register (PCIBAR5MSK)



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

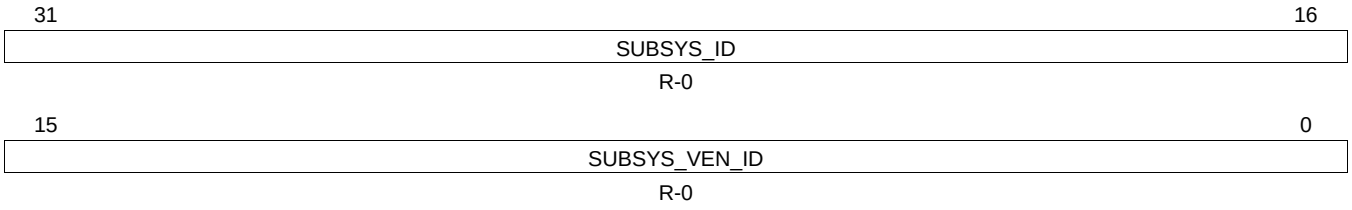
Table 41. Base Address *n* Mask Registers (PCIBAR0MSK-PCIBAR5MSK) Field Descriptions

Bit	Field	Value	Description
31-4	ADDRMASK	0-FFF FFFFh	Address mask bits. These bits control the writeability of the corresponding bits in the corresponding base address *n* mirror register (PCIBAR*n*MIR).
3	PREFETCH_EN	0-1	Prefetchable enable bit. Specifies whether or not the slave memory window controlled by the corresponding base address *n* mirror register (PCIBAR*n*MIR) is prefetchable. This bit is reflected in the PREFETCH bit in the corresponding PCIBAR*n*MIR.
2-0	Reserved	0	Reserved

5.2.9 Subsystem Vendor Identification/Subsystem Identification Mirror Register (PCISUBIDMIR)

The subsystem vendor identification/subsystem identification mirror register (PCISUBIDMIR) is shown in Figure 48 and described in Table 42.

Figure 48. Subsystem Vendor Identification/Subsystem Identification Mirror Register (PCISUBIDMIR)



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

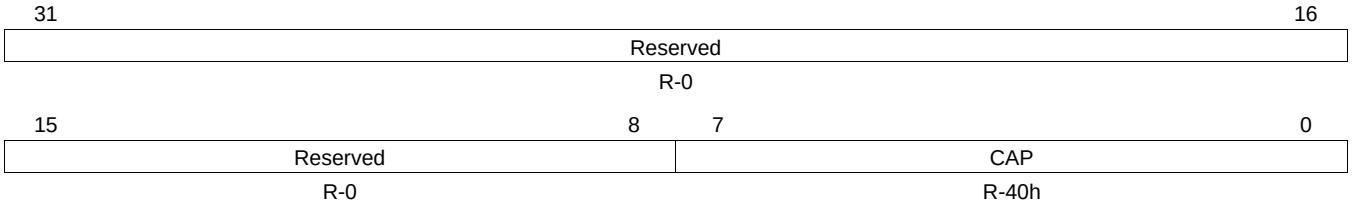
Table 42. Subsystem Vendor Identification/Subsystem Identification Mirror Register (PCISUBIDMIR) Field Descriptions

Bit	Field	Value	Description
31-16	SUBSYS_ID	0-FFFFh	Subsystem ID bits. Identifies the board level device. The Subsystem ID is specified by the board level manufacturer.
15-0	SUBSYS_VEN_ID	0-FFFFh	Subsystem Vendor ID bits. Identifies the board level manufacturer. The Subsystem Vendor ID is specified by the PCI Special Interest Group.

5.2.10 Capabilities Pointer Mirror Register (PCICPBPTRMIR)

The capabilities pointer mirror register (PCICPBPTRMIR) is shown in Figure 49 and described in Table 43.

Figure 49. Capabilities Pointer Mirror Register (PCICPBPTRMIR)



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

Table 43. Capabilities Pointer Mirror Register (PCICPBPTRMIR) Field Descriptions

Bit	Field	Value	Description
31-8	Reserved	0	Reserved
7-0	CAP	40h	Capabilities pointer bits. Specifies the address in configuration space where the first entry in the capabilities list is located.

5.2.11 Maximum Latency/Minimum Grant/Interrupt Pin/Interrupt Line Mirror Register (PCILGINTMIR)

The maximum latency/minimum grant/interrupt pin/interrupt line mirror register (PCILGINTMIR) is shown in Figure 50 and described in Table 44.

Figure 50. Maximum Latency/Minimum Grant/Interrupt Pin/Interrupt Line Mirror Register (PCILGINTMIR)

31	24	23	16
MAX_LAT		MIN_GRNT	
R/W-0		R/W-0	

15	8	7	0
INT_PIN		INT_LINE	
R-1		R/W-0	

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

Table 44. Maximum Latency/Minimum Grant/Interrupt Pin/Interrupt Line Mirror Register (PCILGINTMIR) Field Descriptions

Bit	Field	Value	Description
31-24	MAX_LAT	0-FFh	Maximum latency bits. Specifies how often the device needs to gain access to the PCI bus in 0.25 µsec units.
23-16	MIN_GRNT	0-FFh	Minimum grant bits. Specifies the length of the burst period for the device in 0.25 µsec units.
15-8	INT_PIN	1	Interrupt pin bits. Specifies the interrupt pin the device uses. This bit is hardwired to 01h in the PCI to indicate that interrupt A ($\overline{\text{PINTA}}$) will be used.
7-0	INT_LINE	0-FFh	Interrupt line bits. This value is written by the host and indicates to which input of the system interrupt controller the PCI interrupt pin is connected.

5.2.12 Slave Control Register (PCISLVCNTL)

The slave control register (PCISLVCNTL) is shown in Figure 51 and described in Table 45.

Figure 51. Slave Control Register (PCISLVCNTL)

31	30	29	28	27	26	25	24
Reserved							
R-0							

23	22	21	20	19	18	17	16
Reserved		BASE5_EN	BASE4_EN	BASE3_EN	BASE2_EN	BASE1_EN	BASE0_EN
R-0		R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

15	14	13	12	11	10	9	8
Reserved							

7	6	5	4	3	2	1	0
Reserved			FORCE_DEL_READ_MUL	FORCE_DEL_READ_LN	FORCE_DEL_READ	DIS_SLV_TOUT	CONFIG_DONE
R-0			R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

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

Table 45. Slave Control Register (PCISLVCNTL) Field Descriptions

Bit	Field	Value	Description
31-22	Reserved	0	Reserved
21-16	BASE*n*_EN		Base address enable bits. This bit enables/disables the corresponding base address *n* register. When BASE*n*_EN is cleared to 0 (disabled) any host accesses targeted at the slave memory window covered by the base address register is ignored. Host writes to the base address registers are not affected by BASE*n*_EN.
		0	Disable base address *n* register.
		1	Enable base address *n* register.
15-5	Reserved	0	Reserved
4	FORCE_DEL_READ_MUL		Force Delayed Read Multiple bit.
		0	Slave should respond with normal 16 clock cycle timeout and retry mechanism for Memory Read Multiple transactions
		1	Slave should immediately respond with a retry whenever a Memory Read Multiple transaction is decoded for this slave. This bit overrides the DIS_SLV_TOUT bit for Memory Read Multiple transactions.
3	FORCE_DEL_READ_LN		Force Delayed Read Line bit.
		0	Slave should respond with normal 16 clock cycle timeout and retry mechanism for Memory Read Line transactions
		1	Slave should immediately respond with a retry whenever a Memory Read Line transaction is decoded for this slave. This bit overrides the DIS_SLV_TOUT bit for Memory Read Line transactions.
2	FORCE_DEL_READ		Force Delayed Read bit.
		0	Slave should respond with normal 16 clock cycle timeout and retry mechanism for Memory Read transactions
		1	Slave should immediately respond with a retry whenever a Memory Read transaction is decoded for this slave. This bit overrides DIS_SLV_TOUT for Memory Read transactions.
1	DIS_SLV_TOUT		Disable Slave timeout bit.
		0	Slave responds with normal 16 clock cycle timeout mechanism
		1	Slave will insert wait states on the PCI bus indefinitely until the access is ready to complete
0	CONFIG_DONE		Configuration done bit. Indicates if the configuration registers have been loaded with their proper reset values. Although this bit defaults to 0, the on-chip ROM Boot Loader will set this bit to 1. See Section 2.12.3 for more details.
		0	Configuration registers are being loaded. No access allowed into PCI interface.
		1	Configuration registers loading is complete. PCI interface will accept accesses.

5.2.13 Slave Base Address Translation Registers (PCIBAR0TRL-PCIBAR5TRL)

The slave base address translation registers (PCIBAR0TRL-PCIBAR5TRL) control the translation of transaction addresses as they flow from the PCI bus to the DSP. The translation registers are programmed with a value that replaces the most significant portion of the PCI address as it is converted to a DSP address. The slave base address translation registers are shown in Figure 52 through Figure 57 and described in Table 46.

Figure 52. Slave Base Address 0 Translation Register (PCIBAR0TRL)

31	16
TRANS_ADDR	
R/W-00A0h	

15	4	3	0
TRANS_ADDR		Reserved	
R/W-000h		R-0	

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

Figure 53. Slave Base Address 1 Translation Register (PCIBAR1TRL)

31	16
TRANS_ADDR	
R/W-0180h	

15	4	3	0
TRANS_ADDR		Reserved	
R/W-000h		R-0	

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

Figure 54. Slave Base Address 2 Translation Register (PCIBAR2TRL)

31	16
TRANS_ADDR	
R/W-0204h	

15	4	3	0
TRANS_ADDR		Reserved	
R/W-900h		R-0	

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

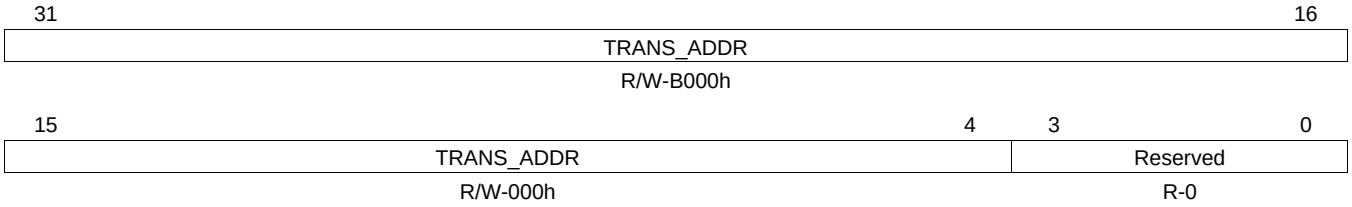
Figure 55. Slave Base Address 3 Translation Register (PCIBAR3TRL)

31	16
TRANS_ADDR	
R/W-A000h	

15	4	3	0
TRANS_ADDR		Reserved	
R/W-000h		R-0	

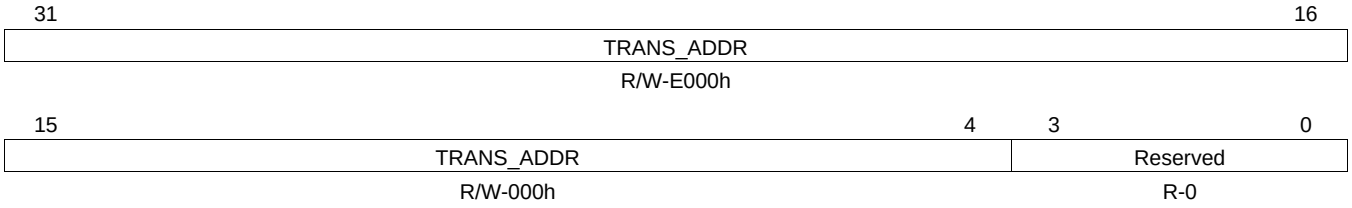
LEGEND: R/W = Read/Write; R = Read only; -*n* = value after reset

Figure 56. Slave Base Address 4 Translation Register (PCIBAR4TRL)



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

Figure 57. Slave Base Address 5 Translation Register (PCIBAR5TRL)



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

Table 46. Slave Base Address *n* Translation Registers (PCIBAR0TRL-PCIBAR5TRL) Field Descriptions

Bit	Field	Value	Description
31-4	TRANS_ADDR	0-FFF FFFFh	Translation address bits. These address bits replace the address bits of a PCI slave transaction. The ADDRMASK bits of the base address *n* mask registers (PCIBAR*n*MSK) specify which bits are replaced in the original PCI address. See Section 3.3 for more details.
3-0	Reserved	0	Reserved

5.2.14 Base Address Mirror Registers (PCIBAR0MIR-PCIBAR5MIR)

The base address mirror registers (PCIBAR0MIR-PCIBAR5MIR) are shown in Figure 58 through Figure 63 and described in Table 47.

Figure 58. Base Address 0 Mirror Register (PCIBAR0MIR)

31	16
ADDR[1]	
R/W-0	

15	4	3	2	1	0
ADDR		PREFETCH	TYPE		IOMEM_SP_IND
R/W-0		R-1	R-0		R-0

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

(1) The access type of this field depends on the address mask bits (ADDRMASK) in the base address 0 mask register (PCIBAR00MSK).

Figure 59. Base Address 1 Mirror Register (PCIBAR1MIR)

31	16
ADDR[1]	
R/W-0	

15	4	3	2	1	0
ADDR		PREFETCH	TYPE		IOMEM_SP_IND
R/W-0		R-0	R-0		R-0

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

(1) The access type of this field depends on the address mask bits (ADDRMASK) in the base address 1 mask register (PCIBAR1MSK).

Figure 60. Base Address 2 Mirror Register (PCIBAR2MIR)

31	16
ADDR[1]	
R/W-0	

15	4	3	2	1	0
ADDR		PREFETCH	TYPE		IOMEM_SP_IND
R/W-0		R-0	R-0		R-0

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

(1) The access type of this field depends on the address mask bits (ADDRMASK) in the base address 2 mask register (PCIBAR2MSK).

Figure 61. Base Address 3 Mirror Register (PCIBAR3MIR)

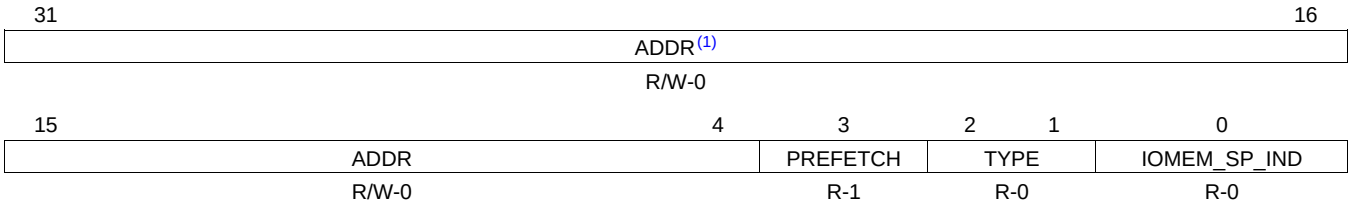
31	16
ADDR[1]	
R/W-0	

15	4	3	2	1	0
ADDR		PREFETCH	TYPE		IOMEM_SP_IND
R/W-0		R-1	R-0		R-0

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

(1) The access type of this field depends on the address mask bits (ADDRMASK) in the base address 3 mask register (PCIBAR3MSK).

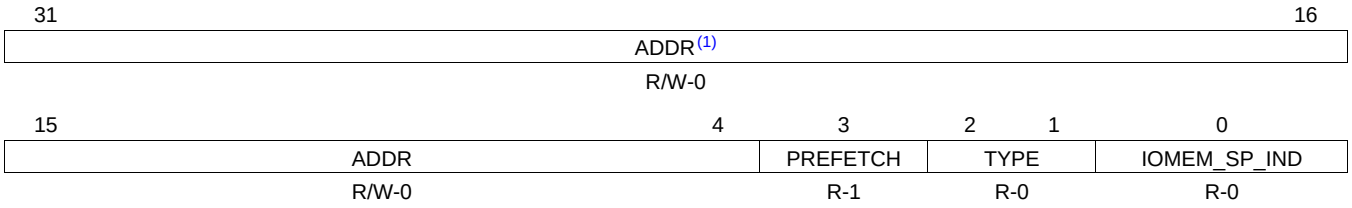
Figure 62. Base Address 4 Mirror Register (PCIBAR4MIR)



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

(1) The access type of this field depends on the address mask bits (ADDRMASK) in the base address 4 mask register (PCIBAR4MSK).

Figure 63. Base Address 5 Mirror Register (PCIBAR5MIR)



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

(1) The access type of this field depends on the address mask bits (ADDRMASK) in the base address 5 mask register (PCIBAR5MSK).

Table 47. Base Address *n* Mirror Registers (PCIBAR0MIR-PCIBAR5MIR) Field Descriptions

Bit	Field	Value	Description
31-4	ADDR	0-FFF FFFFh	Address bits. These bits can be written by the host to allow initialization of the base address at startup. The writeability of individual bits is determined by the corresponding bit in the base address *n* mask register (PCIBAR*n*MSK).
3	PREFETCH	0-1	Prefetchable bit. Specifies whether or not the memory space controlled by this base address register is prefetchable. This bit reflects the value that is input from the PREFETCH_EN bit in the base address *n* mask register (PCIBAR*n*MSK).
2-1	TYPE	0	Type bits. Indicates the size of the base address register/decoder. This version of the PCI only supports 32-bit addressing, so these bits are hardwired to 0.
0	IOMEM_SP_IND	0	IO/Memory Space Indicator bit. Indicates whether the base address maps into the host's memory or I/O space. This version of the PCI only supports memory-mapped base address registers, so this bit is hardwired to 0.

5.2.15 Master Configuration/IO Transaction Proxy Registers

The master configuration/IO access registers cause the PCI to generate configuration or IO transactions. By using an indirect access method (or proxy), the entire configuration and IO space can be addressed, which otherwise would be impossible. For write transactions, the software should write to the data register, then the address register, and then the command register. For read transactions, the software should write to the address register and then the command register. The transaction starts when the command register is written.

5.2.15.1 Master Configuration/IO Access Data Register (PCIMCFGDAT)

When requesting a configuration or IO access, software either writes the data into the master configuration/IO access data register (PCIMCFGDAT) for a write or reads the data from PCIMCFGDAT for a read. The PCIMCFGDAT is shown in Figure 64 and described in Table 48.

Figure 64. Master Configuration/IO Access Data Register (PCIMCFGDAT)

31	0
DATA	
R/W-0	

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

Table 48. Master Configuration/IO Access Data Register (PCIMCFGDAT) Field Descriptions

Bit	Field	Value	Description
31-0	DATA	0-FFFF FFFFh	Data bits. Software writes the data into this register for a configuration/IO write or reads the data from this register for a configuration/IO read.

5.2.15.2 Master Configuration/IO Access Address Register (PCIMCFGADR)

When requesting a configuration or IO access, software writes the desired address for the transaction into the master configuration/IO access address register (PCIMCFGADR). The PCIMCFGADR is shown in Figure 65 and described in Table 49.

Figure 65. Master Configuration/IO Access Address Register (PCIMCFGADR)

31	0
ADDR	
R/W-0	

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

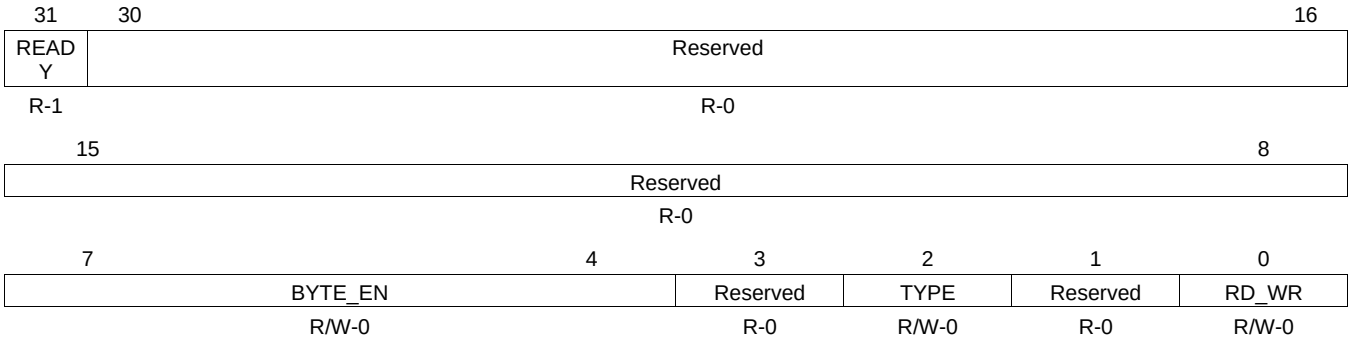
Table 49. Master Configuration/IO Access Address Register (PCIMCFGADR) Field Descriptions

Bit	Field	Value	Description
31-0	ADDR	0-FFFF FFFFh	Address bits. The address bits provide the address for configuration/IO transactions. Bits 1-0 are ignored for a configuration transaction. Software must ensure that bits 1-0 of the address correspond to the BYTE_EN bits in the master configuration/IO access command register (PCIMCFGCMD) for IO transactions, thus ensuring that the access is valid on the PCI bus.

5.2.15.3 Master Configuration/IO Access Command Register (PCIMCFGCMD)

The DSP will program the master configuration/IO access command register (PCIMCFGCMD) to start a configuration read or write. The READY bit should be checked to determine if the previous transaction is complete. The PCIMCFGCMD is shown in Figure 66 and described in Table 50.

Figure 66. Master Configuration/IO Access Command Register (PCIMCFGCMD)



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

Table 50. Master Configuration/IO Access Command Register (PCIMCFGCMD) Field Descriptions

Bit	Field	Value	Description
31	READY		Ready bit. The READY bit should be checked to determine if the previous transaction is complete.
		0	Register is not ready to accept a new command.
		1	Register is ready to accept a new command.
30-8	Reserved	0	Reserved
7-4	BYTE_EN	0-Fh	Byte enable bits. Determines which bytes within the addressed DWORD are being accessed. Byte enables indicate the size of the transfer and must be consistent with bits 1-0 of the address that is specified in the master configuration/IO access address register (PCIMCFGADR).
3	Reserved	0	Reserved
2	TYPE		Type bit. Sets configuration or IO transaction.
		0	Configuration transaction
		1	IO transaction
1	Reserved	0	Reserved
0	RD_WR		Read/write bit. Set read or write operation.
		0	Write
		1	Read

5.2.16 Master Configuration Register (PCIMSTCFG)

The master configuration register (PCIMSTCFG) is shown in Figure 67 and described in Table 51.

Figure 67. Master Configuration Register (PCIMSTCFG)

31	16
Reserved	
R-0	

15	11	10	9	8
Reserved		CFG_FLUSH_IF_NOT_ENABLED	IO_FLUSH_IF_NOT_ENABLED	MEM_FLUSH_IF_NOT_ENABLED
R-0		R/W-0	R/W-0	R/W-0

7	3	2	1	0
Reserved		SW_MEM_RD_MULT_EN	SW_MEM_RD_LINE_EN	SW_MEM_RD_WRINV_EN
R-0		R/W-1	R/W-1	R/W-1

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

Table 51. Master Configuration Register (PCIMSTCFG) Field Descriptions

Bit	Field	Value	Description
31-11	Reserved	0	Reserved
10	CFG_FLUSH_IF_NOT_ENABLED		Configuration flush bit. Controls whether or not the PCI will flush configuration transactions from the proxy registers, if the PCI is not enabled as a master (BUS_MS = 0 in command/status register).
		0	PCI does not flush configuration transactions from the proxy registers.
		1	PCI flushes configuration transactions from the proxy registers.
9	IO_FLUSH_IF_NOT_ENABLED		IO flush bit. Controls whether or not the PCI will flush I/O transactions from the proxy registers, if the PCI is not enabled as a master (BUS_MS = 0 in command/status register).
		0	PCI does not flush I/O transactions from the proxy registers.
		1	PCI flushes I/O transactions from the proxy registers.
8	MEM_FLUSH_IF_NOT_ENABLED		Memory flush bit. Controls whether or not the PCI will flush transactions on the PCIM interface, if the PCI is not enabled as a master (BUS_MS = 0 in command/status register).
		0	PCI does not flush transactions on the PCIM interface.
		1	PCI flushes transactions on the PCIM interface.
7-3	Reserved	0	Reserved
2	SW_MEM_RD_MULT_EN		Memory read multiple enable bit. Controls whether or not the PCI command generation logic is permitted to use the Memory Read Multiple command.
		0	PCI will not generate Memory Read Multiple transactions.
		1	PCI is enabled to generate Memory Read Multiple transactions for appropriate length bursts.
1	SW_MEM_RD_LINE_EN		Memory read line enable bit. Controls whether or not the PCI command generation logic is permitted to use the Memory Read Line command.
		0	PCI will not generate Memory Read Line transactions.
		1	PCI is enabled to generate Memory Read Line transactions for appropriate length bursts.
0	SW_MEM_WRINV_EN		Memory write invalid enable bit. Controls whether or not the PCI command generation logic is permitted to use the Memory Write and Invalidate command.
		0	PCI will not generate Memory Write and Invalidate transactions.
		1	PCI is enabled to generate Memory Write and Invalidate transactions for appropriate length bursts.

5.2.17 PCI Address Substitution Registers (PCIADDSUB0-PCIADDSUB31)

The PCI address substitution registers (PCIADDSUB0-PCIADDSUB31) are used for address translation from the DSP to PCI domain. Using these 32 registers, the master windows can be configured. Each of these 32 registers correspond to 1/32 of the 256 MB addressable PCI space. The PCI address substitution registers are shown in Figure 68 and described in Table 52.

Figure 68. PCI Address Substitution *n* Registers (PCIADDSUB0-PCIADDSUB31)

31	23	22	16
ADD_SUBS		Reserved	
R/W-0		R-0	

15	0
Reserved	
R-0	

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

Table 52. PCI Address Substitution *n* Registers (PCIADDSUB0-PCIADDSUB31) Field Descriptions

Bit	Field	Value	Description
31-23	ADD_SUBS	0-1FFh	Address substitution bits. Substitutes the 9 MSBs of the DSP address during DSP-to-PCI transactions. See Section 4.3 for more details.
22-0	Reserved	0	Reserved

5.3 PCI Configuration Hook Registers

Some of the PCI memory-mapped registers (Section 5.2) are connected to configuration hook registers. See Table 53 for the list of configuration hook registers supported. The values in these configuration hook registers are latched to the actual PCI registers on PCI reset. The default values in these configuration hook registers can be overwritten by software. These registers are implemented mainly to support PCI I2C EEPROM auto-initialization, as discussed in Section 2.12.3.

Table 53. PCI Configuration Hook Registers

Offset	Acronym	Register Description	Section
394h	PCIVENDEVPRG	PCI Vendor ID and Device ID Program Register	Section 5.3.1
39Ch	PCICLREVPRG	PCI Class Code and Revision ID Program Register	Section 5.3.2
3A0h	PCISUBIDPRG	PCI Subsystem Vendor ID and Subsystem ID Program Register	Section 5.3.3
3A4h	PCIMAXLGPRG	Maximum Latency and Minimum Grant Program Register	Section 5.3.4
3ACh	PCICFGDONE	Configuration Done Register	Section 5.3.5

5.3.1 PCI Vendor Identification and Device Identification Program Register (PCIVENDEVPRG)

The PCI vendor identification and device identification program register (PCIVENDEVPRG) provides the default values for the vendor ID/device ID mirror register (PCIVENDEVMIR). PCIVENDEVPRG is shown in Figure 69 and described in Table 54.

Figure 69. PCI Vendor Identification and Device Identification Program Register (PCIVENDEVPRG)

31	0
VENDOR_DEVICE_ID_PROG	
R/W-B003 104Ch	

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

Table 54. PCI Vendor Identification and Device Identification Program Register (PCIVENDEVPRG) Field Descriptions

Bit	Field	Value	Description
31-0	VENDOR_DEVICE_ID_PROG	0-FFFF FFFFh	Vendor device ID program bits. Default values for the VEN_ID and DEV_ID bits in the vendor ID/device ID mirror register (PCIVENDEVMIR).

5.3.2 PCI Class Code and Revision Identification Program Register (PCICLREVPRG)

The PCI class code and revision identification program register (PCICLREVPRG) provides default values for the class code/revision ID mirror register (PCICLREVMIR). PCICLREVPRG is shown in Figure 70 and described in Table 55.

Figure 70. PCI Class Code and Revision Identification Program Register (PCICLREVPRG)

31	0
CLASS_CODE_REV_ID_PROG	
R/W-1180 0001h	

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

Table 55. PCI Class Code and Revision Identification Program Register (PCICLREVPRG) Field Descriptions

Bit	Field	Value	Description
31-0	CLASS_CODE_REV_ID_PROG	0-FFFF FFFFh	Class code revision ID program bits. Default values for the CL_CODE and REV_ID bits in the class code/revision ID mirror register (PCICLREVMIR).

5.3.3 PCI Subsystem Vendor Identification and Subsystem Identification Program Register (PCISUBIDPRG)

The PCI subsystem vendor identification and subsystem identification program register (PCISUBIDPRG) provides default values for the subsystem vendor ID/subsystem ID mirror register (PCISUBIDMIR). PCISUBIDPRG is shown in Figure 71 and described in Table 56.

Figure 71. PCI Subsystem Vendor Identification and Subsystem Identification Program Register (PCISUBIDPRG)

31	0
SUBSYS_VENDOR_ID_SUBSYS_ID_PROG	
R/W-0	

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

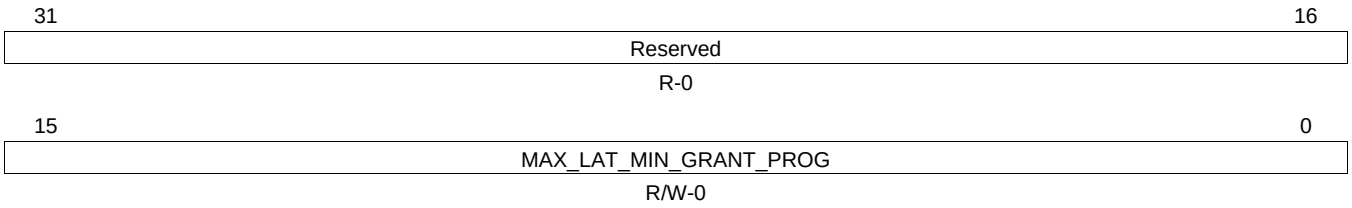
Table 56. PCI Subsystem Vendor Identification and Subsystem Identification Program Register (PCISUBIDPRG) Field Descriptions

Bit	Field	Value	Description
31-0	SUBSYS_VENDOR_ID_SUBSYS_ID_PROG	0-FFFF FFFFh	Subsystem vendor ID and subsystem ID program bits. Default values for the SUBSYS_VEN_ID and SUBSYS_ID bits in the subsystem vendor ID/subsystem ID mirror register (PCISUBIDMIR).

5.3.4 Maximum Latency and Minimum Grant Program Register (PCIMAXLGPRG)

The maximum latency and minimum grant program register (PCIMAXLGPRG) provides default values for the max latency/min grant/interrupt pin/interrupt line mirror register (PCILGINTMIR). PCIMAXLGPRG is shown in Figure 72 and described in Table 57.

Figure 72. Maximum Latency and Minimum Grant Program Register (PCIMAXLGPRG)



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

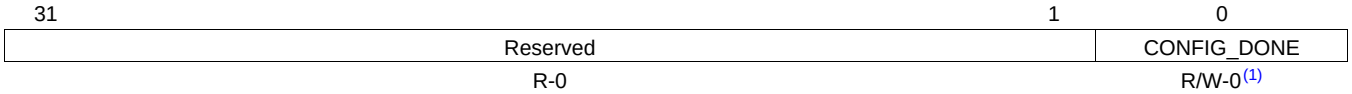
Table 57. Maximum Latency and Minimum Grant Program Register (PCIMAXLGPRG) Field Descriptions

Bit	Field	Value	Description
31-16	Reserved	0	Reserved
15-0	MAX_LAT_MIN_GRANT_PROG	0-FFFFh	Maximum latency minimum grant program bits. Default values for the MAX_LAT and MIN_GRNT bits in the maximum latency/minimum grant/interrupt pin/interrupt line mirror register (PCILGINTMIR).

5.3.5 Configuration Done Register (PCICFGDONE)

Some of the PCI registers can be auto-initialized by the on-chip ROM Boot Loader after reset. When auto-initialization is selected, accesses to the PCI are held off until the CONFIG_DONE bit in the PCI configuration done register (PCICFGDONE) is 1. After initialization is done, the on-chip ROM Boot Loader will set this bit to 1. If auto-initialization is not enabled, the on-chip Boot Loader will set this bit to 1. PCICFGDONE is shown in Figure 73 and described in Table 58.

Figure 73. Configuration Done Register (PCICFGDONE)



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

(1) Although this bit defaults to 0, the on-chip Boot Loader will set this bit to 1.

Table 58. Configuration Done Register (PCICFGDONE) Field Descriptions

Bit	Field	Value	Description
31-1	Reserved	0	Reserved
0	CONFIG_DONE		Configuration done bit. Holds off accesses by external hosts to the PCI until configuration by the on-chip ROM Boot Loader is complete.
		0	Configuration in progress, accesses to the PCI are not permitted.
		1	Configuration complete, accesses to the PCI are allowed.

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 TI 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. Information of third parties may be subject to additional restrictions.

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.

TI products are not authorized for use in safety-critical applications (such as life support) where a failure of the TI product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Buyers represent that they have all necessary expertise in the safety and regulatory ramifications of their applications, and acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of TI products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by TI. Further, Buyers must fully indemnify TI and its representatives against any damages arising out of the use of TI products in such safety-critical applications.

TI products are neither designed nor intended for use in military/aerospace applications or environments unless the TI products are specifically designated by TI as military-grade or "enhanced plastic." Only products designated by TI as military-grade meet military specifications. Buyers acknowledge and agree that any such use of TI products which TI has not designated as military-grade is solely at the Buyer's risk, and that they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

TI products are neither designed nor intended for use in automotive applications or environments unless the specific TI products are designated by TI as compliant with ISO/TS 16949 requirements. Buyers acknowledge and agree that, if they use any non-designated products in automotive applications, TI will not be responsible for any failure to meet such requirements.

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
RFID	www.ti-rfid.com	Telephony	www.ti.com/telephony
Low Power Wireless	www.ti.com/lpw	Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

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