# IDT
# Tsi340™ PCI-to-PCI Bridge
# User Manual

**80E3000_MA001_05**

September 2009

6024 Silver Creek Valley Road, San Jose, California 95138
Telephone: (800) 345-7015 • (408) 284-8200 • FAX: (408) 284-2775
Printed in U.S.A.

GENERAL DISCLAIMER

Integrated Device Technology, Inc. reserves the right to make changes to its products or specifications at any time, without notice, in order to improve design or performance and to supply the best possible product. IDT does not assume any responsibility for use of any circuitry described other than the circuitry embodied in an IDT product. The Company makes no representations that circuitry described herein is free from patent infringement or other rights of third parties which may result from its use. No license is granted by implication or otherwise under any patent, patent rights or other rights, of Integrated Device Technology, Inc.

CODE DISCLAIMER

Code examples provided by IDT are for illustrative purposes only and should not be relied upon for developing applications. Any use of the code examples below is completely at your own risk. IDT MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND CONCERNING THE NONINFRINGEMENT, QUALITY, SAFETY OR SUITABILITY OF THE CODE, EITHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. FURTHER, IDT MAKES NO REPRESENTATIONS OR WARRANTIES AS TO THE TRUTH, ACCURACY OR COMPLETENESS OF ANY STATEMENTS, INFORMATION OR MATERIALS CONCERNING CODE EXAMPLES CONTAINED IN ANY IDT PUBLICATION OR PUBLIC DISCLOSURE OR THAT IS CONTAINED ON ANY IDT INTERNET SITE. IN NO EVENT WILL IDT BE LIABLE FOR ANY DIRECT, CONSEQUENTIAL, INCIDENTAL, INDIRECT, PUNITIVE OR SPECIAL DAMAGES, HOWEVER THEY MAY ARISE, AND EVEN IF IDT HAS BEEN PREVIOUSLY ADVISED ABOUT THE POSSIBILITY OF SUCH DAMAGES. The code examples also may be subject to United States export control laws and may be subject to the export or import laws of other countries and it is your responsibility to comply with any applicable laws or regulations.

LIFE SUPPORT POLICY

Integrated Device Technology's products are not authorized for use as critical components in life support devices or systems unless a specific written agreement pertaining to such intended use is executed between the manufacturer and an officer of IDT.

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

# Contents

# Figures

# Tables

# About this Document

This section discusses the following topics:



## Scope

The *Tsi340 PCI-to-PCI Bridge User Manual* discusses the features, capabilities, and configuration requirements for the Tsi340. It is intended for hardware and software engineers who are designing system interconnect applications with the device.

## Document Conventions

This document uses the following conventions.

### Non-differential Signal Notation

Non-differential signals are either active-low or active-high. An active-low signal has an active state of logic 0 (or the lower voltage level), and is denoted by a lowercase "_b". An active-high signal has an active state of logic 1 (or the higher voltage level), and is not denoted by a special character. The following table illustrates the non-differential signal naming convention.

| State | Single-line signal | Multi-line signal |
|---|---|---|
| Active low | NAME_b | NAME_b[3] |
| Active high | NAME | NAME[3] |

### Object Size Notation

- A *byte* is an 8-bit object.
- A *word* is a 16-bit object.
- A *doubleword* (Dword) is a 32-bit object.

### Numeric Notation

- Hexadecimal numbers are denoted by the prefix *0x* (for example, 0x04).

- Binary numbers are denoted by the prefix *0b* (for example, 0b010).
- Registers that have multiple iterations are denoted by {x..y} in their names; where *x* is first register and address, and *y* is the last register and address. For example, REG{0..1} indicates there are two versions of the register at different addresses: REG0 and REG1.

### Symbols


This symbol indicates a basic design concept or information considered helpful.

This symbol indicates important configuration information or suggestions.

This symbol indicates procedures or operating levels that may result in misuse or damage to the device.

### Document Status Information

- Advance – Contains information that is subject to change, and is available once prototypes are released to customers.
- Preliminary – Contains information about a product that is near production-ready, and is revised as required.
- Formal – Contains information about a final, customer-ready product, and is available once the product is released to production.

## Revision History

### 80E3000_MA001_05, Formal, September 2009

This version of the document was rebranded as IDT. It does not contain any technical changes.

### 80E3000_MA001_02, Advance, October 2006

This version includes numerous minor changes.

### 80E3000_MA001_01, Draft, July 2006

This is the first version of the *Tsi384 User Manual*.

# 1. Functional Overview

This chapter discusses the following topics about the Tsi340:



## 1.1 Overview

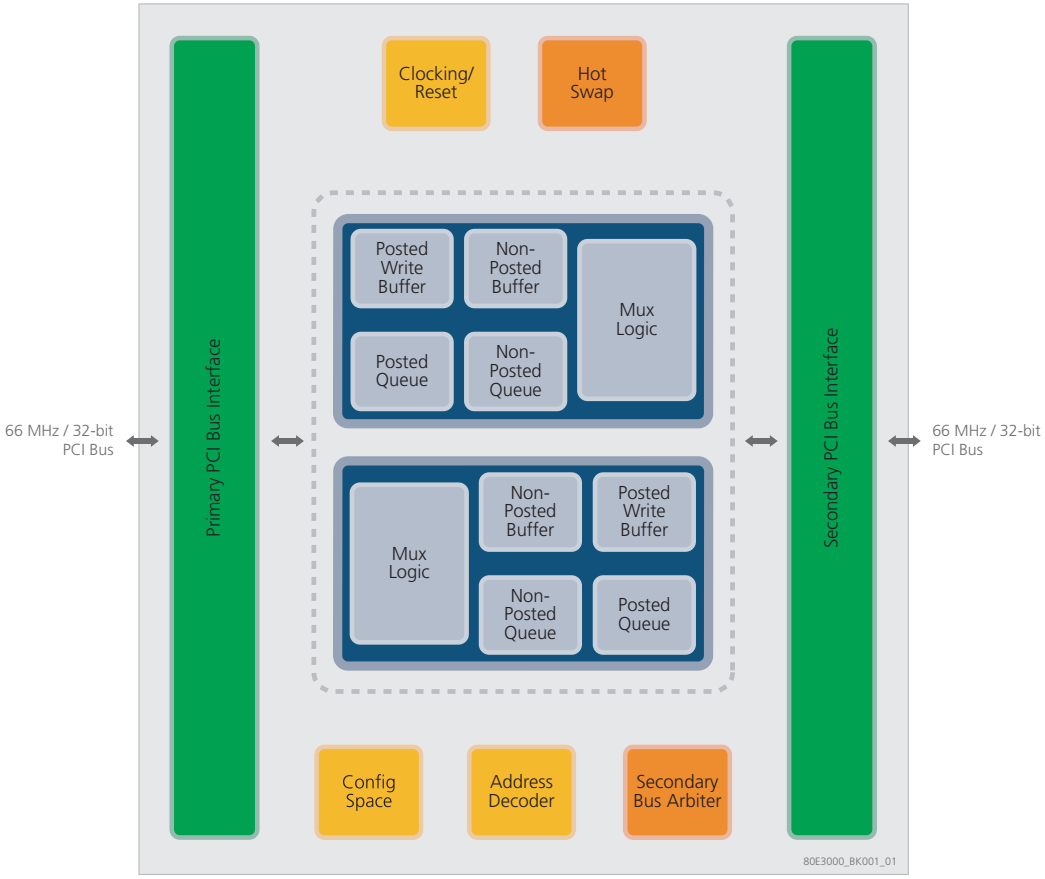
The IDT Tsi340 is a PCI-to-PCI bridge that is fully compliant with the *PCI Local Bus Specification, Revision 2.3*.

The Tsi340 has two identical PCI interfaces that support PCI transactions for each bus. The interfaces can act as either a bus master or a bus slave, depending on the type of transaction.

The Tsi340 enables two PCI buses to operate concurrently. This means that a master and a target on the same PCI bus can communicate while the other PCI bus is busy. This traffic isolation can increase system performance in applications such as multimedia.

The block diagram for Tsi340 is shown Figure 1 on page 16.

**Figure 1: Tsi340 Block Diagram**




## *Typical Applications*

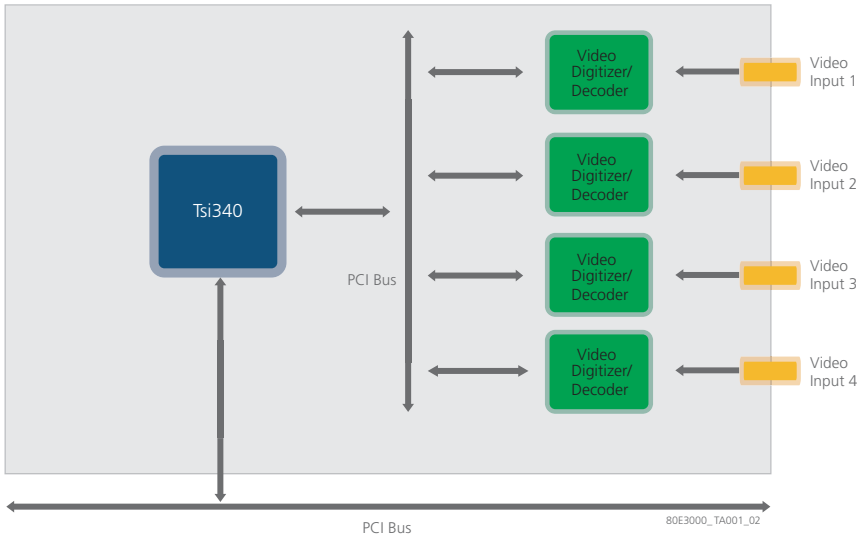
The Tsi340 is suited to applications that need to bridge from PCI to other downstream PCI devices. Its flexibility, and small footprint, make it ideal for a wide range of applications, including:

- Video capture cards
- Digital video recorders
- Industrial PC (IPC) backplanes
- Multi-function printers
- Storage host bus adapters (HBAs)
- Network interface cards (NICs)
- Firewall and security gateways
- Printers, graphics and imaging systems

Option card designers can use the Tsi340 to implement multiple-device PCI option cards. The *PCI Local Bus Specification* loading rules limit PCI option cards to a single connection per PCI signal in the option card connector. Without a PCI-to-PCI bridge, PCI loading rules limit option cards to one device.The Tsi340 overcomes this restriction by providing an independent PCI bus that can support up to four devices.

The application diagram below shows how the Tsi340 enables the design of a multi-component option card and expands the PCI architecture.

**Figure 2: Application Diagram - Digital Video Recorder**




## 1.2 Features

The following section describe the features of Tsi340:

- Industry-standard PCI-to-PCI bridge
- 66 MHz, 32-bit operation on the primary and secondary interfaces
- Up to four PCI bus masters supported on the secondary interface
- Concurrent operation of primary and secondary interfaces
- Compliant with the following specifications:
  - PCI-to-PCI Bridge Architecture Specification (Revision 1.1)
  - PCI Local Bus Specification (Revision 2.3)
  - PCI bus Power Management Interface Specification (Revision 1.1)
  - Advanced Configuration Power Interface (ACPI)
- Posted write buffers in both directions
- 1-KB transaction buffer (total)

- Enhanced address decoding
- Compatible with existing PCI bridging devices from PLX and Pericom
- Physical
  - 128-pin PQFP
  - RoHS compliant
  - 3.3 V I/O, 5 V tolerant
- Compliance
  - *PCI-to-PCI Bridge Architecture Specification (Revision 1.1)*
  - *PCI Local Bus Specification (Revision 2.3)*
  - *PCI bus Power Management Interface Specification (Revision 1.1)*
  - *Advanced Configuration Power Interface (ACPI)*
  - *PICMG CompactPCI Hot-Swap Specification (Revision 2.0)*

## 1.3 Functional Overview

Tsi340 has two PCI interfaces: a primary interface and a secondary interface. Each interface controls the PCI protocol for its respective bus. These interfaces transfer data/control information to and from the Buffer Logic Unit (BLU). The BLU consists of a posted write buffer, posted write queue, non-posted buffer, and non-posted queue.

### 1.3.1 Posted Write Buffer

The Tsi340 handles the conventional PCI transactions of Memory Write, and Memory Write and Invalidate as posted transactions.

The posted write buffer is used for temporary storage of data flowing from the primary interface to the secondary interface and from the secondary interface to the primary interface. Each posted buffer has a capacity of 256 bytes. The amount of space assigned to each transaction is dynamic. A single transaction can use sizes ranging from one memory location (4 bytes) to 64-memory location (64 bytes).

When the Tsi340 determines that a memory write transaction must be forwarded across the bridge, it first checks for empty space in the posted write buffer. If space is available, the posted write buffer accepts data until the buffer is full or the transaction is terminated. If there is no space in the posted write buffer, the transaction is terminated with retry.

### 1.3.2 Posted Write Queue

The posted write queue is used to store the control information related to the transaction flowing from the primary interface to secondary interface or from the secondary interface to the primary interface. Each posted write queue has a four entry FIFO, which provides four active posted write transactions in each direction. Data related to each entry is stored in the posted write buffer.

The posted write queue accepts an entry from an external master as long as at least one entry is free and at least one Dword of space is available in the posted write buffer.

### 1.3.3 Non-Posted Buffer

The non-posted buffer is used for storing the data related to delayed transactions. The following list of transactions use the non-posted buffer:

- Memory Read
- Memory Read Line
- Memory Read Multiple
- I/O Read
- I/O Write
- Type-1 Configuration Read
- Type-1 Configuration Write

All the non-posted transactions are processed through the non-posted queues and non-posted buffers. Each non-posted buffer has a storage capacity of up to 256 bytes for storing data related to delayed transactions.

### 1.3.4 Non-Posted Queue

The non-posted queue is used to store the control information related to the all non-posted transactions. Each non-posted queue has a four entry FIFO, which provides four active non-posted transactions in each direction. Data related to each entry is stored in the non-posted buffer.

The non-posted queue accepts an entry from an external master if at least one entry is available. If all four entries are full the Tsi340 retries the external master until an entry becomes available.

### 1.3.5 Configuration Space

Tsi340 is a PCI-to-PCI bridge, and complies with the *PCI to PCI Bridge Architecture Specification, Revision 1.1*. The Tsi340's configuration space can only be accessed from the primary interface. The Tsi340 uses additional device specific configuration registers to support optional, device specific features.

Refer to "Configuration Transactions" on page 33 for more information.

### 1.3.6 Address Decoding Logic

The Tsi340 is a transparent bridge. In transparent mode, the I/O, Memory, pre-fetchable memory base and limit, and optional base address registers 0 and 1 define address ranges residing on the secondary bus. All other addresses are assumed to reside on the primary bus. Inverse address decoding determines when to forward the transaction up-stream.

Refer to "Address Decoding" on page 45 for more information.

### 1.3.7 Secondary Bus Arbiter

The Tsi340 has an internal secondary bus arbiter. It provides bus arbitration for up to four additional masters. Each external master is assigned to either high or low priority, or may be masked off.

The internal arbiter provides a two level arbitration scheme in which arbitration is divided into the following two groups: a high priority group and low priority group. Each master can be assigned to either high priority group or low priority group through the configuration register.

Refer to “PCI Bus Arbitration” on page 59 for more information.

### 1.3.8 Hot Swap Interface

Tsi340 is designed with an interface for Hot Swap support. This allows the user to insert or extract the bridge card without powering down the system. During insertion and extraction process, the bridge indicates to system software about the Hot Swap event by driving HS_ENUM_b. It also provides a visual indication through the HS_LED_OUT signal.

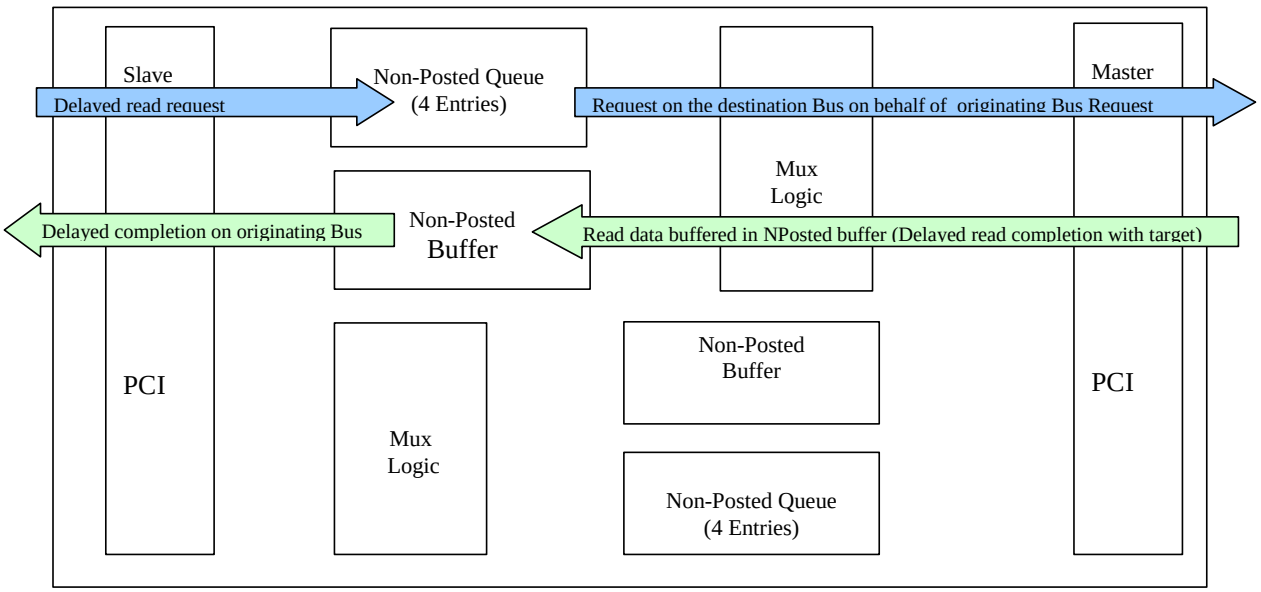
## 1.4 Data Flow

The following sections describe the data flow through the Tsi340 device.

### 1.4.1 Memory Read Transactions

The conventional PCI memory read, memory read line, and memory read multiple commands are used to transfer memory read data. Tsi340 completes all memory read transactions as delayed transactions. Figure 3 shows the Tsi340 memory read flow.

**Figure 3: Memory Read Flow**



The following steps detail the memory read flow through the Tsi340:

- The read request from the initiator is posted/entered into the Non-Posted Queue,
- The transaction is terminated by signaling target retry to the initiator
- When the target retry is received, the initiator is required to continue to repeat the same read transaction until at least one data transfer is completed or until a master/target abort is received
- The Tsi340 then arbitrates for the destination bus and initiates a read transaction using the exact read address and read command
  - If Tsi340 receives retry on the target bus, it continues to repeat the read transaction until at least one data transfer is completed, or until an error condition is encountered.

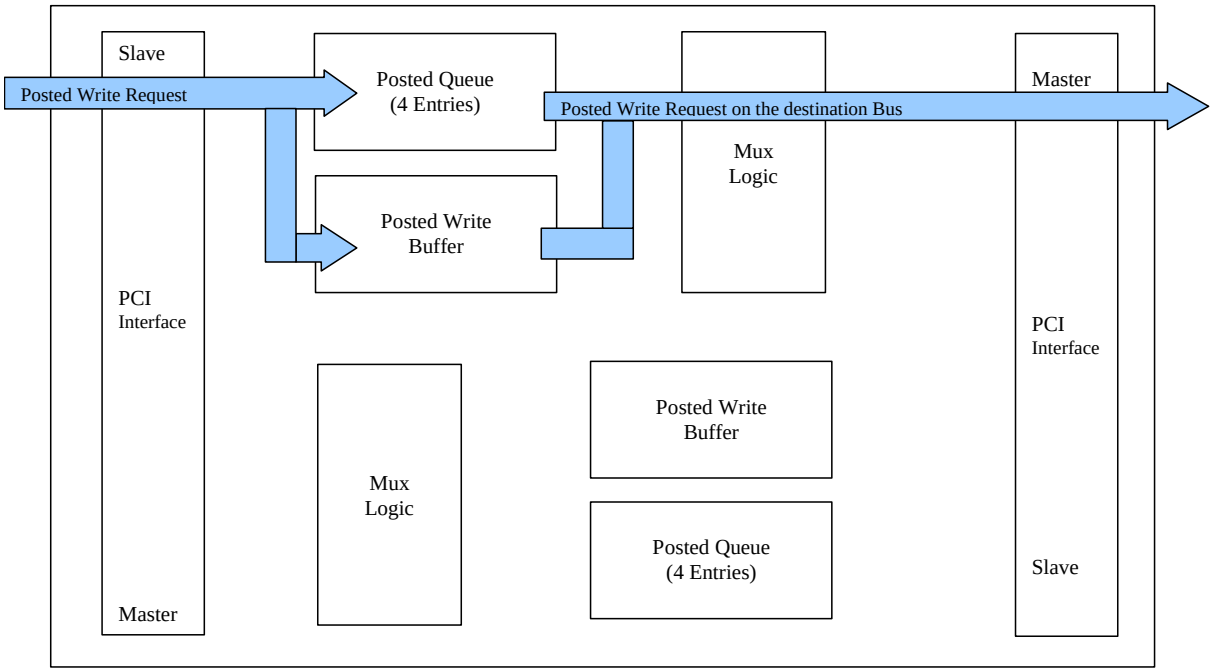
When a memory read transaction targets non-prefetchable address space the Tsi340 will pre-fetch a single DWORD of data when the memory read command is used. For all other read transactions the Tsi340 will pre-fetch data according to the pre-fetch algorithm, see “Read Transactions” on page 29.

- When the transaction is completed on the target bus, the Tsi340 transfers the data to the initiator when the initiator repeats the transaction

### 1.4.2 Posted Write Transaction Flow

The conventional PCI memory write and memory write and invalidate are posted transactions. Unlike non-posted transactions these transactions are first completed on the originating bus and then completed on the destination bus. Figure 4 shows the Tsi340 posted write flow.

**Figure 4: Memory Write Flow**




The following steps detail the posted write flow through the Tsi340:

- When Tsi340 determines that a memory write transaction is to be forwarded across the bridge, it first checks for empty space in the posted write buffer
  - If space is available, Tsi340 accepts data until the buffer is full or the transaction is terminated.
  - If there is no space in the posted write buffer, the transaction is terminated with retry.
- After buffering data into the posted buffer the Tsi340 arbitrates for the destination bus and writes the data to the destination.

# 2. PCI Interface

This chapter discusses the following topics about the Tsi340:



## 2.1 Overview

Tsi340 has two PCI interfaces: a primary interface and a secondary interface. Each interface controls the PCI protocol for its respective bus. These interfaces transfer data/control information to and from the Buffer Logic Unit (BLU). The BLU consists of a posted write buffer, posted write queue, non-posted buffer, and non-posted queue.

The following sections describe the how the Tsi340 handles PCI transactions, transaction forwarding across Tsi340, and transaction termination.

## 2.2 Transaction Types

This section provides a summary of PCI transactions performed by Tsi340. Table 1 lists the command code and name of each PCI transaction. The Master and Target columns indicate Tsi340 support for each transaction when Tsi340 initiates transactions as a master, on the primary bus and on the secondary bus, and when Tsi340 responds to transactions as a target, on the primary bus and on the secondary bus.

**Table 1: Type of Transactions**

| Type of Transaction | Initiates as a Master | | Responds as a Target | |
|---|---|---|---|---|
| | Primary | Secondary | Primary | Secondary |
| 0000-Interrupt Acknowledge | No | No | No | No |
| 0001-Special Cycle | Yes | Yes | No | No |
| 0010-I/O Read | Yes | Yes | Yes | Yes |
| 0011-I/O Write | Yes | Yes | Yes | Yes |
| 0100-Reserved | No | No | No | No |

**Table 1: Type of Transactions**

| Type of Transaction | Initiates as a Master | | Responds as a Target | |
|---|---|---|---|---|
| | Primary | Secondary | Primary | Secondary |
| 0101-Reserved | No | No | No | No |
| 0110-Memory Read | Yes | Yes | Yes | Yes |
| 0111-Memory Write | Yes | Yes | Yes | Yes |
| 1000-Reserved | No | No | No | No |
| 1001-Reserved | No | No | No | No |
| 1010-Configuration read | No | Yes | Yes | No |
| 1011-Configuration Write | Type-1 | Yes | Yes | Type-1 |
| 1100-Memory Read Multiple | Yes | Yes | Yes | Yes |
| 1101-Dual Address Cycle | Yes | Yes | Yes | Yes |
| 1110-Memory Read Line | Yes | Yes | Yes | Yes |
| 1111-Memory Write and Invalidate | Yes | Yes | Yes | Yes |

### 2.2.1 Transaction Types Not Supported

As indicated in Table 1 the following PCI commands are not supported by Tsi340:

- Tsi340 never initiates a PCI transaction with a reserved command code and, as a target, Tsi340 ignores reserved command codes.
- Tsi340 never initiates an interrupt acknowledge transaction and, as a target, Tsi340 ignores interrupt acknowledge transactions. Interrupt acknowledge transactions are expected to reside entirely on the primary PCI bus closest to the host bridge.
- Tsi340 does not respond to special cycle transactions. Tsi340 cannot guarantee delivery of a special cycle transaction to downstream buses because of the broadcast nature of the special cycle command and the inability to control the transaction as a target. To generate special cycle transactions on other PCI buses, either upstream or downstream, a type-1 configuration command must be used.
- Tsi340 does not generate Type 0 configuration transactions on the primary interface, nor does it respond to Type 0 configuration transactions on the secondary PCI interface. The *PCI-to-PCI Bridge Architecture Specification* does not support configuration from the secondary bus.

### 2.2.2 Address Phase

A standard PCI transaction consists of one or two address phases, followed by one or more data phases. The first address phase is designated by an asserting (falling) edge on the FRAME# signal. The number of address phases depends on whether the address is 32 bits or 64 bits.

#### 2.2.2.1 Single Address Phase

A 32-bit address uses a single address phase. This address is driven on AD[31:0], and the bus command is driven on C/BE#[3:0]. Tsi340 supports the linear increment address mode only for decoding memory address space, which is indicated when the lower two address bits are equal to 0. If either of the lower two address bits is nonzero, Tsi340 automatically disconnects the transaction after the first data transfer.

#### 2.2.2.2 Dual Address Phase

Dual address transactions are PCI transactions that contain two address phases specifying a 64-bit address. The first address phase is denoted by the asserting edge of FRAME#. The second address phase always follows on the next clock cycle.

For a 32-bit interface, the first address phase contains the dual address command code on the C/BE#<3:0> lines, and the low 32 address bits on the AD<31:0> lines. The second address phase consists of the specific memory transaction command code on the C/BE#<3:0> lines and the high 32 address bits on the AD<31:0>lines. In this way, 64-bit addressing can be supported on 32-bit PCI buses.

The *PCI-to-PCI Bridge Architecture Specification* supports the use of dual address transactions in the prefetchable memory range only. Tsi340 supports dual address transactions in both the upstream and the downstream direction. Tsi340 supports a programmable 64-bit address range in prefetchable memory for downstream forwarding of dual address transactions. Dual address transactions falling outside the prefetchable address range are forwarded upstream, but not downstream. Prefetching and posting are performed in a manner consistent with the guidelines given in this specification for each type of memory transaction in prefetchable memory space.

Tsi340 responds only to dual address transactions that use the following transaction command codes:

- Memory Write
- Memory Write and Invalidate
- Memory Read
- Memory Read Line
- Memory Read Multiple

Use of other transaction codes may result in a master abort.

Any memory transactions addressing the first 4GB space should use a single address phase; that is, the high 32 bits of a dual address transaction should never be 0.

### 2.2.3 Device Select (DEVSEL#) Generation

Tsi340 always performs positive address decoding when accepting transactions on either the primary or secondary buses. Tsi340 never subtractively decodes. Medium DEVSEL# timing is used on both interfaces.

## 2.2.4 Data Phase Transactions

The address phase or phases of a PCI transaction are followed by one or more data phases. A data phase is completed when IRDY# and either TRDY# or STOP# are asserted. A transfer of data occurs only when both IRDY# and TRDY# are asserted during the same PCI clock cycle. The last data phase of a transaction is indicated when FRAME# is deasserted and both TRDY# and IRDY# are asserted, or when IRDY# and STOP# are asserted.

Depending on the command type, Tsi340 can support multiple data phase PCI transactions.

### 2.2.4.1 Write Transactions

Write transactions are treated as either posted write or delayed write transactions.

#### *Posted Write Transactions*

Posted write forwarding is used for memory write and for memory write and invalidate transactions.

When Tsi340 determines that a memory write transaction is to be forwarded across the bridge, Tsi340 asserts DEVSEL# with medium timing and TRDY# in the next cycle, provided that enough buffer space is available in the posted data queue for the address and at least 8 words of data. This enables Tsi340 to accept write data without obtaining access to the target bus. Tsi340 can accept one Dword of write data every PCI clock cycle; that is, no target wait states are inserted. This write data is stored in internal posted write buffers and is subsequently delivered to the target.

Tsi340 continues to accept write data until one of the following events occurs:

- The initiator terminates the transaction by deasserting FRAME# and IRDY#
- An internal write address boundary is reached, such as a cache line boundary or an aligned 4 KB boundary, depending on the transaction type
- The posted write data buffer is full

When one of the last two events occurs, Tsi340 returns a target disconnect to the requesting initiator on this data phase to terminate the transaction. Once the posted write data moves to the head of the posted data queue, Tsi340 asserts its request on the target bus. This can occur while the Tsi340 is still receiving data on the initiator bus. When the grant for the target bus is received and the target bus is detected in the idle condition, Tsi340 asserts FRAME# and drives the stored write address out on the target bus. On the following cycle, Tsi340 drives the first Dword of write data and continues to transfer write data until all write data corresponding to that transaction is delivered, or until a target termination is received. As long as write data exists in the queue, Tsi340 can drive 1 Dword of write data each PCI clock cycle.

Tsi340 ends the transaction on the target bus when one of the following conditions is met:

- All posted write data has been delivered to the target
- The target returns a target disconnect or target retry (Tsi340 starts another transaction to deliver the rest of the write data)
- The target returns a target abort (Tsi340 discards remaining write data).
- The master latency timer expires, and Tsi340 no longer has the target bus grant (Tsi340 starts another transaction to deliver remaining write data).

### Memory Write and Invalidate Transactions

Posted write forwarding is used for memory write and invalidate transactions. Memory write and invalidate transactions guarantee transfer of entire cache lines. If the write buffer fills before an entire cache line is transferred, Tsi340 disconnects the transaction and converts it to a memory write transaction.

Tsi340 disconnects memory write and invalidate transactions at aligned cache line boundaries. The cache line size value in Tsi340 cache line size register provides the number of Dwords in a cache line. For Tsi340 to generate memory write and invalidate transactions, this cache line size value must be written to a value that is a nonzero power of 2 and less than or equal to 16 (that is, 1, 2, 4, 8, or 16 Dwords).

If the cache line size does not meet the memory write and invalidate conditions, that is, the value is 0, or is not a power of 2, or is greater than 16 Dwords, Tsi340 treats the memory write and invalidate command as a memory write command. In this case, when Tsi340 forwards the memory write and invalidate transaction to the target bus, it converts the command code to a memory write code and does not observe cache line boundaries.

If the value in the cache line size register does meet the memory write and invalidate conditions, that is, the value is a nonzero power of 2 less than or equal to 16 Dwords, Tsi340 returns a target disconnect to the initiator either on a cache line boundary or when the posted write buffer fills. For a cache line size of 16 Dwords, Tsi340 disconnects a memory write and invalidate transaction on every cache line boundary. When the cache line size is 1, 2, 4, or 8 Dwords, Tsi340 accepts another cache line if at least 8 Dwords of empty space remains in the posted write buffer. If less than 8 Dwords of empty space remains, Tsi340 disconnects on that cache line boundary. When the memory write and invalidate transaction is disconnected before a cache line boundary is reached, typically because the posted write buffer fills, the transaction is converted to a memory write transaction.

### Delayed Write Transactions

Delayed write forwarding is used for I/O write transactions and for Type 1 configuration write transactions.

A delayed write transaction guarantees that the actual target response is returned back to the initiator without holding the initiating bus in wait states. A delayed write transaction is limited to a single Dword data transfer.

When a write transaction is first detected on the initiator bus, and Tsi340 forwards it as a delayed transaction, Tsi340 claims the access by asserting DEVSEL# and returns a target retry to the initiator. During the address phase, Tsi340 samples the bus command, address, and address parity one cycle later. After IRDY# is asserted, Tsi340 also samples the first data Dword, byte enable bits, and data parity. This information is placed into the delayed transaction queue. The transaction is queued only if no other existing delayed transactions have the same address and command, and if the delayed transaction queue is not full. When the delayed write transaction moves to the head of the delayed transaction queue and all ordering constraints with posted data are satisfied, Tsi340 initiates the transaction on the target bus. Tsi340 transfers the write data to the target. If Tsi340 receives a target retry in response to the write transaction on the target bus, it continues to repeat the write transaction until the data transfer is completed, or until an error condition is encountered.

If Tsi340 is unable to deliver write data after $2^{24}$ attempts, Tsi340 ceases further write attempts and returns a target abort to the initiator. The delayed transaction is removed from the delayed transaction queue. Tsi340 also asserts P_SERR_b if the primary SERR# enable bit is set in the command register (see “Primary Command Register—Offset 0x04” on page 108).

When the initiator repeats the same write transaction (same command, address, byte enable bits, and data), and the completed delayed transaction is at the head of the queue, Tsi340 claims the access by asserting DEVSEL# and returns TRDY# to the initiator, to indicate that the write data was transferred. If the initiator requests multiple Dwords, Tsi340 also asserts STOP# in conjunction with TRDY# to signal a target disconnect. Note that only those bytes of write data with valid byte enable bits are compared. If any of the byte enable bits are turned off (driven high), the corresponding byte of write data is not compared.

If the initiator repeats the write transaction before the data has been transferred to the target, Tsi340 returns a target retry to the initiator. Tsi340 continues to return a target retry to the initiator until write data is delivered to the target, or until an error condition is encountered. When the write transaction is repeated, Tsi340 does not make a new entry into the delayed transaction queue.

Tsi340 implements a discard timer that starts counting when the delayed write completion is at the head of the delayed transaction queue. The initial value of this timer can be set to one of two values, selectable through both the primary and secondary master time-out bits in the bridge control register (see “Bridge Control Register—Offset 0x3C” on page 122). If the initiator does not repeat the delayed write transaction before the discard timer expires, Tsi340 discards the delayed write transaction from the delayed transaction queue. Tsi340 also conditionally asserts P_SERR_b.

### *Write Transaction Address Boundaries*

Tsi340 imposes internal address boundaries when accepting write data. The aligned address boundaries are used to prevent Tsi340 from continuing a transaction over a device address boundary and to provide an upper limit on maximum latency. Tsi340 returns a target disconnect to the initiator when it reaches the aligned address boundaries under the conditions shown in Table 2.

**Table 2: Posted Write Address Boundaries**

| Type of Transaction | Condition | Aligned Address Boundary |
|---|---|---|
| Memory write | Disconnect control bit = 0 (“Chip Control Register/Diagnostic Control — Offset 0x44” on page 126) | 4 KB aligned address boundary |
| Memory write | Disconnect control bit = 1 (“Chip Control Register/Diagnostic Control — Offset 0x44” on page 126) | Disconnects at $n^{th}$ cache line boundary |
| Memory write and Invalidate | Cache line size ~=1, 2, 4, 8, 16 (“Cache Line Size Register—Offset 0x0C” on page 112) | 4 KB aligned address boundary |

**Table 2: Posted Write Address Boundaries**

| Type of Transaction | Condition | Aligned Address Boundary |
|---|---|---|
| Memory write and invalidate | Cache line size = 1, 2, 4, 8 ("Cache Line Size Register—Offset 0x0C" on page 112) | $n^{th}$ cache line boundary, where a cache line boundary is reached and less than 8 free D-words of posted write buffer space remains |
| Memory write and invalidate | Cache line size = 16 ("Cache Line Size Register—Offset 0x0C" on page 112) | 16-Dword aligned address boundary |

#### *Buffering Multiple Write Transactions*

The memory write disconnect control bit is located in the chip control register at offset 40h in configuration space. Tsi340 continues to accept posted memory write transactions as long as space for at least 1 Dword of data in the posted write data buffer remains. If the posted write data buffer fills before the initiator terminates the write transaction, Tsi340 returns a target disconnect to the initiator.

Delayed write transactions are handled as long as at least one open entry in Tsi340 delayed transaction queue exists. Therefore, several posted and delayed write transactions can exist in data buffers at the same time.

#### *Fast Back-to-Back Write Transactions*

Tsi340 can recognize fast back-to-back write transactions as a target on the PCI bus. When Tsi340 cannot accept the second transaction because of buffer space limitations, it returns a target retry to the initiator.

Tsi340 does not perform write combining or merging.

### 2.2.4.2 Read Transactions

Delayed read forwarding is used for all read transactions crossing theTsi340. Delayed read transactions are treated as either prefetchable or nonprefetchable.

Table 3 shows the read behavior, prefetchable or non-prefetchable, for each type of read operation.

**Table 3: Read Behavior**

| Type of Transaction | Read Behavior |
|---|---|
| I/O read | Prefetching never done |
| Configuration read | Prefetching never done |
| Memory read | Downstream: Prefetching is used if the address falls within a prefetchable region. If the address falls within a non-prefetchable region, 1 DWORD is fetched.<br>Upstream: Prefetching used if prefetch disable is off (default) |

**Table 3: Read Behavior**

| Type of Transaction | Read Behavior |
|---|---|
| Memory read line | Prefetching always used |
| Memory read multiple | Prefetching always used |

### Prefetchable Read Transactions

A prefetchable read transaction is a read transaction where Tsi340 performs speculative Dword reads, transferring data from the target before it is requested from the initiator. This behavior allows a prefetchable read transaction to consist of multiple data transfers. However, byte enable bits cannot be forwarded for all data phases as is done for the single data phase of the nonprefetchable read transaction. For prefetchable read transactions, Tsi340 forces all byte enable bits to be turned on for all data phases.

Prefetchable behavior is used for memory read line and memory read multiple transactions, as well as for memory read transactions that fall into prefetchable memory space.

The amount of data that is prefetched depends on the type of transaction. The amount of prefetching may also be affected by the amount of free buffer space available in Tsi340, and by any read address boundaries encountered.

Prefetching should not be used for those read transactions that have side effects in the target device, that is, control and status registers, FIFOs, and so on. The target device's base address register or registers indicate if a memory address region is prefetchable.

### Nonprefetchable Read Transactions

A nonprefetchable read transaction is a read transaction where Tsi340 requests 1–and only 1–Dword from the target and disconnects the initiator after delivery of the first Dword of read data. Unlike prefetchable read transactions, Tsi340 forwards the read byte enable information for the data phase.

Nonprefetchable behavior is used for I/O and configuration read transactions, as well as for memory read transactions that fall into nonprefetchable memory space. If prefetching could have side effects, for example, when accessing a FIFO, use nonprefetchable read transactions to those locations. Accordingly, if it is important to retain the value of the byte enable bits during the data phase, use nonprefetchable read transactions. If these locations are mapped in memory space, use the memory read command and map the target into nonprefetchable (memory-mapped I/O) memory space to utilize nonprefetching behavior.

### Read Prefetch Address Boundaries

Tsi340 imposes internal read address boundaries on read prefetching. When a read transaction reaches one of these aligned address boundaries, Tsi340 stops prefetching data, unless the target signals a target disconnect before the read prefetch boundary is reached. When Tsi340 finishes transferring this read data to the initiator, it returns a target disconnect with the last data transfer, unless the initiator completes the transaction before all prefetched read data is delivered. Any leftover prefetched data is discarded.

Prefetchable read transactions in flow-through mode prefetch to the nearest aligned 4KB address boundary, or until the initiator deasserts FRAME#.

Table 4 shows the read prefetch address boundaries for read transactions during non-flow-through mode.

**Table 4: Prefetch Address Boundaries**

| Type of Transaction | Address Space | Cache Line Size | Prefetch Aligned Address Boundary |
|---|---|---|---|
| Configuration read | -- | -- | 1 Dword (no prefetch) |
| I/O read | -- | -- | 1 Dword (no prefetch) |
| Memory read | Nonprefetchable | -- | 1 Dword (no prefetch) |
| Memory read | Prefetchable | CLS ≠ 1, 2, 4, 8 | 16-Dword aligned address boundary |
| Memory read | Prefetchable | CLS = 1, 2, 4, 8 | Cache line address boundary |
| Memory read line | -- | CLS ≠ 1, 2, 4, 8 | 16-Dword aligned address boundary |
| Memory read line | -- | CLS = 1, 2, 4, 8 | Cache line address boundary |
| Memory read multiple | -- | CLS ≠ 1, 2, 4, 8 | Queue full |
| Memory read multiple | -- | CLS = 1, 2, 4, 8 | Second cache line boundary |

### Delayed Read Requests

The Tsi340 treats all read transactions as delayed read transactions, which means that the read request from the initiator is posted into a delayed transaction queue. Read data from the target is placed in the read data queue directed toward the initiator bus interface and is transferred to the initiator when the initiator repeats the read transaction.

When the Tsi340 accepts a delayed read request, it first samples the read address, read bus command, and address parity. When IRDY# is asserted, the Tsi340 then samples the byte enable bits for the first data phase. This information is entered into the delayed transaction queue.

The Tsi340 terminates the transaction by signaling a target retry to the initiator. Upon reception of the target retry, the initiator is required to continue to repeat the same read transaction until at least one data transfer is completed, or until a target response other than a target retry (target abort, or master abort) is received.

### Delayed Read Completion with Target

When the delayed read request reaches the head of the delayed transaction queue, and all previously queued posted write transactions have been delivered, Tsi340 arbitrates for the target bus and initiates the read transaction, using the exact read address and read command captured from the initiator during the initial delayed read request. If the read transaction is a nonprefetchable read, Tsi340 drives the captured byte enable bits during the next cycle. If the transaction is a prefetchable read transaction, it drives the captured byte enables for first data phase and drives all byte enable bits to 0 for all remaining data phases.

If Tsi340 receives a target retry in response to the read transaction on the target bus, it continues to repeat the read transaction until at least one data transfer is completed, or until an error condition is encountered. If the transaction is terminated through normal master termination or target disconnect after at least one data transfer has been completed, Tsi340 does not initiate any further attempts to read more data.

If Tsi340 is unable to obtain read data from the target after $2^{24}$ attempts, Tsi340 ceases further read attempts and returns a target abort to the initiator. The delayed transaction is removed from the delayed transaction queue. Tsi340 also asserts P_SERR_b if the primary SERR# enable bit is set in the command register.

Once Tsi340 receives DEVSEL# and TRDY# from the target, it transfers the data read to the opposite direction read data queue, pointing toward the opposite interface, before terminating the transaction. For example, read data in response to a downstream read transaction initiated on the primary bus is placed in the upstream read data queue. Tsi340 can accept 1 Dword of read data each PCI clock cycle; that is, no master wait states are inserted. The number of Dwords transferred during a delayed read transaction depends on the conditions given in Table 4 (assuming no disconnect is received from the target).

### Delayed Read Completion on Initiator Bus

When the transaction has been completed on the target bus, and the delayed read data is at the head of the read data queue, and all ordering constraints with posted write transactions have been satisfied, Tsi340 transfers the data to the initiator when the initiator repeats the transaction. For memory read transactions, Tsi340 aliases the memory read, memory read line, and memory read multiple bus commands when matching the bus command of the transaction to the bus command in the delayed transaction queue. Tsi340 returns a target disconnect along with the transfer of the last Dword of read data to the initiator. If the initiator terminates the transaction before all read data has been transferred, the remaining read data left in data buffers is discarded.

When the master repeats the transaction and starts transferring prefetchable read data from Tsi340 data buffers while the read transaction on the target bus is still in progress and before a read boundary is reached on the target bus, the read transaction starts operating in flow-through mode. Because data is flowing through the data buffers from the target to the initiator, long read bursts can then be sustained. In this case, the read transaction is allowed to continue until the initiator terminates the transaction, or until an aligned 4KB address boundary is reached, or until the buffer fills, whichever comes first. When the buffer empties, Tsi340 reflects the stalled condition to the initiator by deasserting TRDY# until more read data is available; otherwise, Tsi340 does not insert any target wait states. When the initiator terminates the transaction, the deassertion of FRAME# on the initiator bus is forwarded to the target bus. Any remaining read data is discarded.

Tsi340 implements a discard timer that starts counting when the delayed read completion is at the head of the delayed transaction queue, and the read data is at the head of the read data queue. The initial value of this timer can be set to one of two values, selectable through both the primary and secondary master time-out value bits in the bridge control register. If the initiator does not repeat the read transaction before the discard timer expires Tsi340 discards the read transaction and the read data from its queues. Tsi340 also conditionally asserts P_SERR_b.

Tsi340 has the capability to post multiple delayed read requests, up to a maximum of three in each direction. If an initiator starts a read transaction that matches the address and read command of a read transaction that is already queued, the current read command is not posted as it is already contained in the delayed transaction queue.

## 2.3 Configuration Transactions

Configuration transactions are used to initialize a PCI system. Every PCI device has a configuration space that is accessed by configuration commands. All Tsi340 registers are accessible in configuration space only.

In addition to accepting configuration transactions for initialization of its own configuration space, Tsi340 also forwards configuration transactions for device initialization in hierarchical PCI systems, as well as for special cycle generation.

Two types of configuration transactions are supported, Type 0 and Type 1.

- Type 0 configuration transactions are issued when the intended target resides on the same PCI bus as the initiator. A Type 0 configuration transaction is identified by the configuration command and the lowest 2 bits of the address set to 00b.
- Type 1 configuration transactions are issued when the intended target resides on another PCI bus, or when a special cycle is to be generated on another PCI bus. A Type 1 configuration command is identified by the configuration command and the lowest 2 address bits set to 01b.

Figure 5 and Figure 6 show the address formats for Type 0 and Type 1 configuration transactions.

**Figure 5: Type 0 Configuration Transaction Address Format**

| 31 11 | 10 8 | 7 2 | 1 | 0 |
|---|---|---|---|---|
| Reserved | Function number | Register number | 0 | 0 |

**Figure 6: Type 1 Configuration Transaction Address Format**

| 31 24 | 23 16 | 15 11 | 10 8 | 7 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| Reserved | Bus number | Device number | Function number | Register number | 0 | 1 |

The register number is found in both Type 0 and Type 1 formats and gives the Dword address of the configuration register to be accessed. The function number is also included in both Type 0 and Type 1 formats and indicates which function of a multifunction device is to be accessed. For single function devices, this value is not decoded. Type 1 configuration transaction addresses also include a 5-bit field designating the device number that identifies the device on the target PCI bus that is to be accessed. In addition, the bus number in Type 1 transactions specifies the PCI bus to which the transaction is targeted.

### 2.3.1 Type 0 Access to Tsi340

Tsi340 configuration space is accessed by a Type 0 configuration transaction on the primary interface. Tsi340 configuration space cannot be accessed from the secondary bus. Tsi340 responds to a Type 0 configuration transaction by asserting P_DEVSEL_b when the following conditions are met during the address phase:

- The bus command is a configuration read or configuration write transaction
- Low 2 address bits P_AD<1:0> must be 00b
- Signal P_IDSEL must be asserted.

The function code is ignored because Tsi340 is a single-function device.

Tsi340 limits all configuration accesses to a single Dword data transfer and returns a target disconnect with the first data transfer if additional data phases are requested. Because read transactions to Tsi340 configuration space do not have side effects, all bytes in the requested Dword are returned, regardless of the value of the byte enable bits.

Type 0 configuration write and read transactions do not use Tsi340 data buffers; that is, these transactions are completed immediately, regardless of the state of the data buffers.

Tsi340 ignores all Type 0 transactions initiated on the secondary interface.

### 2.3.2 Type 1 to Type 0 Translation

Type 1 configuration transactions are used specifically for device configuration in a hierarchical PCI bus system. A PCI-to-PCI bridge, such as the Tsi340, is the only type of device that should respond to a Type 1 configuration command. Type 1 configuration commands are used when the configuration access is intended for a PCI device that resides on a PCI bus other than the one where the Type 1 transaction is generated.

Tsi340 performs a Type 1 to Type 0 translation when the Type 1 transaction is generated on the primary bus and is intended for a device attached directly to the secondary bus. Tsi340 must convert the configuration command to a Type 0 format so that the secondary bus device can respond to it. Type 1 to Type 0 translations are performed only in the downstream direction; that is, Tsi340 generates a Type 0 transaction only on the secondary bus, and never on the primary bus.

Tsi340 forwards Type 1 to Type 0 configuration read or write transactions as delayed transactions. Type 1 to Type 0 configuration read or write transactions are limited to a single 32-bit data transfer.

#### 2.3.2.1 Address Phase Requirements

Tsi340 responds to a Type 1 configuration transaction and translates it into a Type 0 transaction on the secondary bus when the following conditions are met during the address phase:

- The lower two address bits on P_AD<1:0> are 01b.
- The bus number in address field P_AD<23:16> is equal to the value in the secondary bus number register in Tsi340 configuration space.

- The bus command on P_CBE_b<3:0> is a configuration read or configuration write transaction.

When Tsi340 translates the Type 1 transaction to a Type 0 transaction on the secondary interface, it performs the following translations to the address:

- Sets the low 2 address bits on S_AD[1:0] to 00b.
- Decodes the device number and drives the bit pattern specified in Table 5 on S_AD[31:16] for the purpose of asserting the device's IDSEL signal.
- Sets S_AD[15:11] to 0.
- Leaves unchanged the function number and register number fields.

#### 2.3.2.2 Address and Device Mapping

Tsi340 asserts a unique address line based on the device number. These address lines may be used as secondary bus IDSEL signals. The mapping of the address lines depends on the device number in the Type 1 address bits P_AD[15:11]. Table 5 presents the mapping used by the Tsi340.

**Table 5: Device Number to IDSEL S_AD Mapping**

| Device Number | P_AD<15:11> | Secondary IDSEL S_AD<31:16> | S_AD Bit |
|---|---|---|---|
| 0h | 00000 | 0000 0000 0000 0001 | 16 |
| 1h | 00001 | 0000 0000 0000 0010 | 17 |
| 2h | 00010 | 0000 0000 0000 0100 | 18 |
| 3h | 00011 | 0000 0000 0000 1000 | 19 |
| 4h | 00100 | 0000 0000 0001 0000 | 20 |
| 5h | 00101 | 0000 0000 0010 0000 | 21 |
| 6h | 00110 | 0000 0000 0100 0000 | 22 |
| 7h | 00111 | 0000 0000 1000 0000 | 23 |
| 8h | 01000 | 0000 0001 0000 0000 | 24 |
| 9h | 01001 | 0000 0010 0000 0000 | 25 |
| Ah | 01010 | 0000 0100 0000 0000 | 26 |
| Bh | 01011 | 0000 1000 0000 0000 | 27 |
| Ch | 01100 | 0001 0000 0000 0000 | 28 |
| Dh | 01101 | 0010 0000 0000 0000 | 29 |
| Eh | 01110 | 0100 0000 0000 0000 | 30 |
| Fh | 01111 | 1000 0000 0000 0000 | 31 |

**Table 5: Device Number to IDSEL S_AD Mapping**

| Device Number | P_AD<15:11> | Secondary IDSEL S_AD<31:16> | S_AD Bit |
|---|---|---|---|
| 10h-1Eh | 10000-11110 | 0000 0000 0000 0000 | -- |
| 1Fh | 11111 | Generate special cycle (P_AD<7:2> = 00h)<br>0000 0000 0000 0000 (P_AD<7:2> ≠ 00h) | -- |

Tsi340 can assert up to 16 unique address lines to be used as IDSEL signals for up to 16 devices on the secondary bus, for device numbers ranging from 0 through 15. Because of electrical loading constraints of the PCI bus, more than 16 IDSEL signals should not be necessary. However, if device numbers greater than 15 are desired, some external method of generating IDSEL lines must be used, and no upper address bits are then asserted. The configuration transaction is still translated and passed from the primary bus to the secondary bus. If no IDSEL pin is asserted to a secondary device, the transaction ends in a master abort.

### 2.3.3 Type 1 to Type 1 Forwarding

Type 1 to Type 1 transaction forwarding provides a hierarchical configuration mechanism when two or more levels of PCI-to-PCI bridges are used.

Tsi340 forwards Type 1 to Type 1 configuration write transactions as delayed transactions. Type 1 to Type 1 configuration write transactions are limited to a single data transfer.

When Tsi340 detects a Type 1 configuration transaction intended for a PCI bus downstream from the secondary bus, Tsi340 forwards the transaction unchanged to the secondary bus. Ultimately, this transaction is translated to a Type 0 configuration command or to a special cycle transaction by a downstream PCI-to-PCI bridge.

#### 2.3.3.1 Address Phase Requirements

Downstream Type 1 to Type 1 forwarding occurs when the following conditions are met during the address phase:

- The lower two address bits are equal to 01b.
- The bus number falls in the range defined by the lower limit (exclusive) in the secondary bus number register and the upper limit (inclusive) in the subordinate bus number register.
- The bus command is a configuration read or write transaction.

Tsi340 also supports Type 1 to Type 1 forwarding of configuration write transactions upstream to support upstream special cycle generation. A Type 1 configuration command is forwarded upstream when the following conditions are met:

- The lower two address bits are equal to 01b.
- The bus number falls outside the range defined by the lower limit (inclusive) in the secondary bus number register and the upper limit (inclusive) in the subordinate bus number register.
- The device number in address bits AD<15:11> is equal to 11111b.

- The function number in address bits AD<10:8> is equal to 111b.
- The bus command is a configuration write transaction.

Tsi340 forwards Type 1-to-Type 1 configuration write transactions as delayed transactions. Type 1-to-Type 1 configuration write transactions are limited to a single data transfer.

### 2.3.4 Special Cycles

The Type 1 configuration mechanism is used to generate special cycle transactions in hierarchical PCI systems. Special cycle transactions are ignored by a PCI-to-PCI bridge acting as a target and are not forwarded across the bridge. Special cycle transactions can be generated from Type 1 configuration write transactions in either the upstream or the downstream direction.

Tsi340 initiates a special cycle on the target bus when a Type 1 configuration write transaction is detected on the initiating bus and the following conditions are met during the address phase:

- The lower two address bits on AD<1:0> are equal to 01b
- The device number in address bits AD<15:11> is equal to 11111b
- The function number in address bits AD<10:8> is equal to 111b.
- The register number in address bits AD<7:2> is equal to 000000b
- The bus number is equal to the value in the secondary bus number register in configuration space for downstream forwarding or equal to the value in the primary bus number register in configuration space for upstream forwarding
- The bus command on C/BE# is a configuration write command

When Tsi340 initiates the transaction on the target interface, the bus command is changed from a configuration write to a special cycle and the following actions occur:

- The address and data are forwarded unchanged
- Devices that use special cycles ignore the address and decode only the bus command
- The data phase contains the special cycle message
- The transaction is forwarded as a delayed transaction, but in this case the target response is not forwarded back (because special cycles result in a master abort)
- Once the transaction is completed on the target bus, through detection of the master abort condition, Tsi340 responds with TRDY# to the next attempt of the configuration transaction from the initiator.
- If more than one data transfer is requested, Tsi340 responds with a target disconnect operation during the first data phase.

## 2.4 Transaction Termination

This section describes how Tsi340 returns transaction termination conditions back to the initiator.

The initiator can terminate transactions with one of the following types of terminations:

- Normal termination: Normal termination occurs when the initiator deasserts FRAME# at the beginning of the last data phase, and deasserts IRDY# at the end of the last data phase in conjunction with either TRDY# or STOP#
- Master abort: A master abort occurs when no target response is detected. When the initiator does not detect a DEVSEL# from the target within five clock cycles after asserting FRAME#, the initiator terminates the transaction with a master abort. If FRAME# is still asserted, the initiator deasserts FRAME# on the next cycle, and then deasserts IRDY# on the following cycle. IRDY# must be asserted in the same cycle in which FRAME# deasserts. If FRAME# is already deasserted, IRDY# can be deasserted on the next clock cycle following detection of the master abort condition.

The target can terminate transactions with one of the following types of terminations:

- Normal termination: TRDY# and DEVSEL# asserted in conjunction with FRAME# deasserted and IRDY# asserted.
- Target retry: STOP# and DEVSEL# asserted without TRDY# during the first data phase. No data transfers occur during the transaction. This transaction must be repeated.
- Target disconnect with data transfer: STOP# and DEVSEL# asserted with TRDY#. Signals that this is the last data transfer of the transaction.
- Target disconnect without data transfer: STOP# and DEVSEL# asserted without TRDY# after previous data transfers have been made. Indicates that no more data transfers will be made during this transaction.
- Target abort: STOP# asserted without DEVSEL# and without TRDY#. Indicates that the target will never be able to complete this transaction. DEVSEL# must be asserted for at least one cycle during the transaction before the target abort is signaled.

### 2.4.1 Master Termination Initiated by Tsi340

Tsi340, as an initiator, uses normal termination if DEVSEL# is returned by the target within five clock cycles of Tsi340's assertion of FRAME# on the target bus.

As an initiator, Tsi340 terminates a transaction when the following conditions are met:

- During a delayed write transaction, a single Dword is delivered.
- During a non-prefetchable read transaction, a single Dword is transferred from the target.
- During a prefetchable read transaction, a prefetch boundary is reached.
- For a posted write transaction, all write data for the transaction is transferred from Tsi340 data buffers to the target.
- For a burst transfer, with the exception of memory write and invalidate transactions, the master latency timer expires and Tsi340's bus grant is deasserted.
- The target terminates the transaction with a retry, disconnect, or target abort.

#### 2.4.1.1 Master Latency Timer Expiration

If Tsi340 is delivering posted write data when it terminates the transaction because the master latency timer expires, it initiates another transaction to deliver the remaining write data. The address of the transaction is updated to reflect the address of the current Dword to be delivered. If Tsi340 is prefetching read data when it terminates the transaction because the master latency timer expires, it does not repeat the transaction to obtain more data.

### 2.4.2 Master Abort Received by Tsi340

If Tsi340 initiates a transaction on the target bus and does not detect DEVSEL# returned by the target within five clock cycles of Tsi340's assertion of FRAME#, Tsi340 terminates the transaction with a master abort. Tsi340 sets the received master abort bit in the status register corresponding to the target bus.

For delayed read and write transactions, when the master abort mode bit in the "Bridge Control Register—Offset 0x3C" on page 122 is 0, Tsi340 returns TRDY# on the initiator bus and, for read transactions, returns FFFF_FFFFh as data. When the master abort mode bit is 1, Tsi340 returns target abort on the initiator bus. Tsi340 also sets the signaled target abort bit in the register corresponding to the initiator bus.

When a master abort is received in response to a posted write transaction, Tsi340 discards the posted write data and makes no more attempts to deliver the data. Tsi340 sets the received master abort bit in the status register when the master abort is received on the primary bus, or it sets the received master abort bit in the secondary status register when the master abort is received on the secondary interface.

When a master abort is detected in response to a posted write transaction, and the master abort mode bit is set, the Tsi340 also asserts P_SERR_b. The functionality must be enabled by the SERR# enable bit in the command register and not be disabled by the device-specific P_SERR_b disable bit for master abort during posted write transactions (that is, master abort mode = 1; SERR# enable bit = 1; and P_SERR_b disable bit for master aborts = 0.)

When Tsi340 performs a Type 1 to special cycle translation, a master abort is the expected termination for the special cycle on the target bus. In this case, the master abort received bit is *not* set, and the Type 1 configuration transaction is disconnected after the first data phase.

### 2.4.3 Target Termination Received by Tsi340

When Tsi340 initiates a transaction on the target bus and the target responds with DEVSEL#, the target can end the transaction with one of the following types of termination:

- Normal termination (upon deassertion of FRAME#)
- Target retry
- Target disconnect
- Target abort

Tsi340 handles these terminations in different ways, depending on the type of transaction being performed.

### 2.4.4 Delayed Write Target Termination Response

When Tsi340 initiates a delayed write transaction, the type of target termination received from the target can be passed back to the initiator. Table 6 shows Tsi340 response to each type of target termination that occurs during a delayed write transaction.

**Table 6: Tsi340 Response to Delayed Write Target Termination**

| Target Termination | Response |
|---|---|
| Normal | Return disconnect to initiator with first data transfer only if multiple data phases requested. |
| Target retry | Return target retry to initiator. Continue write attempts to target. |
| Target disconnect | Return disconnect to initiator with first data transfer only if multiple data phases requested. |
| Target abort | Return target abort to initiator. Set received target abort bit in target interface status register. Set signaled target abort bit in initiator interface status register. |

Tsi340 repeats a delayed write transaction until one of the following conditions is met:

- Tsi340 completes at least one data transfer
- Tsi340 receives a master abort
- Tsi340 receives a target abort

- Tsi340 makes $2^{24}$ write attempts resulting in a response of target retry
  - — After Tsi340 makes $2^{24}$ attempts of the same delayed write transaction on the target bus, Tsi340 asserts P_SERR_b if the primary SERR# enable bit is set in the command register and the implementation-specific P_SERR_b disable bit for this condition is not set in the P_SERR_b event disable register. Tsi340 stops initiating transactions in response to that delayed write transaction. The delayed write request is discarded. Upon a subsequent write transaction attempt by the initiator, Tsi340 returns a target abort.

### 2.4.5 Posted Write Target Termination Response

When Tsi340 initiates a posted write transaction, the target termination cannot be passed back to the initiator. Table 7 shows Tsi340 response to each type of target termination that occurs during a posted write transaction.

**Table 7: Tsi340 Response to Posted Write Target Termination**

| Target Termination | Response |
|---|---|
| Normal | No additional action. |
| Target retry | Repeat write transaction to target. |
| Target disconnect | Initiate write transaction to deliver remaining posted write data. |
| Target abort | Set received target abort bit in the target interface status register. Assert P_SERR_b if enabled, and set the signaled system error bit in the primary status register. |

When a target retry or target disconnect is returned and posted write data associated with that transaction remains in the write buffers, Tsi340 initiates another write transaction to attempt to deliver the rest of the write data. In the case of a target retry, the same address is driven as for the initial write transaction attempt. If a target disconnect is received, the address that is driven on a subsequent write transaction attempt is updated to reflect the address of the current Dword. If the initial write transaction is a memory write and invalidate transaction, and a partial delivery of write data to the target is performed before a target disconnect is received, Tsi340 uses the memory write command to deliver the rest of the write data because less than a cache line will be transferred in the subsequent write transaction attempt.

After Tsi340 makes $2^{24}$ write transaction attempts and fails to deliver all the posted write data associated with that transaction, Tsi340 asserts P_SERR_b if the primary SERR# enable bit is set in the command register and the device-specific P_SERR_b disable bit for this condition is not set in the P_SERR_b event disable register. The write data is discarded.

### 2.4.6 Delayed Read Target Termination Response

When Tsi340 initiates a delayed read transaction, the abnormal target responses can be passed back to the initiator. Other target responses depend on how much data the initiator requests. Table 8 shows Tsi340 response to each type of target termination that occurs during a delayed read transaction.

**Table 8: Tsi340 Response to Delayed Read Target Termination**

| Target Termination | Response |
|---|---|
| Normal | If prefetchable, target disconnect only if initiator requests more data than read from target. If non prefetchable, target disconnect on first data phase. |
| Target retry | Re-initiate read transaction to target. |
| Target disconnect | If initiator requests more data than read from target, return target disconnect to initiator. |
| Target abort | Return target abort to initiator. Set received target abort bit in the target interface status register.<br>Set signaled target abort bit in the initiator interface status register. |

Tsi340 repeats a delayed read transaction until one of the following conditions is met:

- Tsi340 completes at least one data transfer.
- Tsi340 receives a master abort.
- Tsi340 receives a target abort.
- Tsi340 makes $2^{24}$ read attempts resulting in a response of target retry.
  - After Tsi340 makes $2^{24}$ attempts of the same delayed read transaction on the target bus, Tsi340 asserts P_SERR_b if the primary SERR# enable bit is set in the command register and the implementation-specific P_SERR_b disable bit for this condition is not set in the P_SERR_b event disable register. Tsi340 stops initiating transactions in response to that delayed read transaction. The delayed read request is discarded. Upon a subsequent read transaction attempt by the initiator, Tsi340 returns a target abort.

### 2.4.7 Target Termination Initiated by Tsi340

Tsi340 can return a target retry, target disconnect, or target abort to an initiator for reasons other than detection of that condition at the target interface.

#### 2.4.7.1 Target Retry

Tsi340 returns a target retry to the initiator when it cannot accept write data or return read data as a result of internal conditions.

Tsi340 returns a target retry to an initiator when any of the following conditions are met:

- For delayed write transactions:
  - The transaction is being entered into the delayed transaction queue
  - The transaction has already been entered into the delayed transaction queue, but target response has not yet been received
  - Target response has been received but has not progressed to the head of the return queue
  - The delayed transaction queue is full, and the transaction cannot be queued
  - A transaction with the same address and command has been queued
- For delayed read transactions:
  - The transaction is being entered into the delayed transaction queue
  - The read request has already been queued, but read data is not yet available
  - Data has been read from the target, but it is not yet at the head of the read data queue, or a posted write transaction precedes it
  - The delayed transaction queue is full, and the transaction cannot be queued
  - A delayed read request with the same address and bus command has already been queued
  - Tsi340 is currently discarding previously prefetched read data
- For posted write transactions:
  - The posted write data buffer does not have enough space for address and at least 8 Dwords of write data

##### *Delayed Transaction*

When a target retry is returned to the initiator of a delayed transaction, the initiator must repeat the transaction with the same address and bus command as well as the data if this is a write transaction, within the time frame specified by the master time-out value; otherwise, the transaction is discarded from Tsi340 buffers.

#### 2.4.7.2 Target Disconnect

Tsi340 returns a target disconnect to an initiator when one of the following conditions is met:

- Tsi340 hits an internal address boundary
- Tsi340 cannot accept any more write data

- Tsi340 has no more read data to deliver

#### 2.4.7.3 Target Abort

Tsi340 returns a target abort to an initiator when one of the following conditions is met:

- Tsi340 is returning a target abort from the intended target.
- Tsi340 is unable to obtain delayed read data from the target or to deliver delayed write data to the target after $2^{24}$ attempts.

When Tsi340 returns a target abort to the initiator, it sets the signaled target abort bit in the status register corresponding to the initiator interface.

## 2.5 CompactPCI Hot-swap Support

The Tsi340 is hot-swap friendly silicon that supports all of the hot-swap capable features, contains support for software control, and integrates circuitry required by the *PICMG CompactPCI Hot-Swap Specification.*

- To be hot-swap capable, the Tsi340 supports the following:
- Compliance with *PCI Local Bus Specification.*
- Tolerance of $V_{DD}$ from early power.
- Asynchronous reset.
- Tolerance of precharge voltage.
- I/O buffers that meet modified V/I requirements.
- Limited I/O terminal voltage at pre-charge voltage.
- Hot-swap control and status programming via extended PCI capabilities linked list.
- Hot-swap terminals: HS_ENUM_b, HS_SWITCH_b, and HS_LED, CompactPCI hot-swap defines a process for installing and removing PCI boards without adversely affecting a running system. The Tsi340 provides this functionality such that it can be implemented on a board that can be removed and inserted in a hot-swap system.

Tsi340 provides three terminals to support hot-swap when configured to be in hot-swap mode:

- HS_ENUM_b (output) - Indicates to the system that an insertion event occurred or that a removal event is about to occur.
- HS_SWITCH_b (input) - Indicates the state of a board ejector handle.
- HS_LED_OUT(output) - Drives a blue LED to signal insertion- and removal-ready status.

# 3. Address Decoding

This chapter discusses the following:



## 3.1 Overview of Address Decoding

Tsi340 uses three address ranges that control I/O and memory transaction forwarding. These address ranges are defined by base and limit address registers in Tsi340 configuration space. This chapter describes these address ranges, as well as ISA-mode and VGA-addressing support.

## 3.2 Address Ranges

Tsi340 uses the following address ranges that determine which I/O and memory transactions are forwarded from the primary PCI bus to the secondary PCI bus, and from the secondary PCI bus to the primary PCI bus:

- One 32-bit I/O address range
- One 32-bit memory-mapped I/O (non-prefetchable memory)
- One 64-bit prefetchable memory address range

Transactions falling within these ranges are forwarded downstream from the primary PCI bus to the secondary PCI bus. Transactions falling outside these ranges are forwarded upstream from the secondary PCI bus to the primary PCI bus.

Tsi340 uses a flat address space; that is, it does not perform any address translations. The address space has no *gaps* — addresses that are not marked for downstream forwarding are always forwarded upstream.

## 3.3 I/O Address Decoding

Tsi340 uses the following mechanisms that are defined in Tsi340 configuration space to specify the I/O address space for downstream and upstream forwarding:

- I/O base and limit address registers ("I/O Base Address Register—Offset 0x1C" on page 115)
- The ISA enable bit ("Bridge Control Register—Offset 0x3C" on page 122)
- The legacy ISA I/O enable bit ("Miscellaneous Control Register — Offset 0xC0" on page 140)

- The VGA enable bit (“Bridge Control Register—Offset 0x3C” on page 122)
- The VGA snoop bit (“Primary Command Register—Offset 0x04” on page 108)

This section provides information on the I/O address registers and ISA mode. “VGA Mode” on page 53 provides information on the VGA modes.

To enable downstream forwarding of I/O transactions, the I/O enable bit must be set in the command register in Tsi340 configuration space (see “Primary Command Register—Offset 0x04” on page 108). If the I/O enable bit is not set, all I/O transactions initiated on the primary bus are ignored. To enable upstream forwarding of I/O transactions, the master enable bit must be set in the command register. If the master enable bit is not set, Tsi340 ignores all I/O and memory transactions initiated on the secondary bus. Setting the master enable bit also allows upstream forwarding of memory transactions.

If any Tsi340 configuration state affecting I/O transaction forwarding is changed by a configuration write operation on the primary bus at the same time that I/O transactions are ongoing on the secondary bus, Tsi340 response to the secondary bus I/O transactions is not predictable.

Configure the I/O base and limit address registers, ISA enable bit, VGA enable bit, and VGA snoop bit before setting the I/O enable and master enable bits, and change them subsequently only when the primary and secondary PCI buses are idle.

## 3.3.1 Base and Limit Address Registers

Tsi340 implements one set of I/O base and limit address registers in configuration space that define an I/O address range for downstream forwarding (see “I/O Base Address Register—Offset 0x1C” on page 115). Tsi340 supports 32-bit I/O addressing, which allows I/O addresses downstream of Tsi340 to be mapped anywhere in a 4 GB I/O address space.

I/O transactions with addresses that fall inside the range defined by the I/O base and limit registers are forwarded downstream from the primary PCI bus to the secondary PCI bus. I/O transactions with addresses that fall outside this range are forwarded upstream from the secondary PCI bus to the primary PCI bus.

### 3.3.1.1 Turning off the I/O Range

The I/O range can be turned off by setting the I/O base address to a value greater than that of the I/O limit address. When the I/O range is turned off, all I/O transactions are forwarded upstream, and no I/O transactions are forwarded downstream.

Tsi340 I/O range has a minimum granularity of 4 kB and is aligned on a 4 kB boundary. The maximum I/O range is 4 GB in size.

### 3.3.1.2 I/O Base Register

The I/O base register consists of an 8-bit field at configuration address 1Ch, and a 16-bit field at address 30h. The top 4 bits of the 8-bit field define bits <15:12> of the I/O base address. The bottom 4 bits read only as 1h to indicate that Tsi340 supports 32-bit I/O addressing. Bits <11:0> of the base address are assumed to be 0, which naturally aligns the base address to a 4 kB boundary.

The 16 bits contained in the I/O base upper 16 bits register at configuration offset 30h define AD<31:16> of the I/O base address. All 16 bits are read/write. After primary bus reset or chip reset, the value of the I/O base address is initialized to 0000 0000h.

#### 3.3.1.3 I/O Limit Register

The I/O limit register consists of an 8-bit field at configuration offset 1Dh and a 16-bit field at offset 32h. The top 4 bits of the 8-bit field define bits <15:12> of the I/O limit address. The bottom 4 bits read only as 1h to indicate that 32-bit I/O addressing is supported. Bits <11:0> of the limit address are assumed to be FFFh, which naturally aligns the limit address to the top of a 4kB I/O address block. The 16 bits contained in the I/O limit upper 16 bits register at configuration offset 32h define AD<31:16> of the I/O limit address. All 16 bits are read/write. After primary bus reset or chip reset, the value of the I/O limit address is reset to 0000 0FFFh.

The initial states of the I/O base and I/O limit address registers define an I/O range of 0000 0000h to 0000 0FFFh, which is the bottom 4 kB of I/O space. Write these registers with their appropriate values before either setting the I/O enable bit or the master enable bit in the command register in configuration space.

### 3.3.2 ISA Mode

Tsi340 supports ISA mode by providing an ISA enable bit in the bridge control register in configuration space (“Bridge Control Register—Offset 0x3C” on page 122). ISA mode modifies the response of Tsi340 inside the I/O address range in order to support mapping of I/O space in the presence of an ISA bus in the system. This bit only affects the response of Tsi340 when the transaction falls inside the address range defined by the I/O base and limit address registers, and only when this address also falls inside the first 64 kB of I/O space (address bits <31:16> are 0000h).

When the ISA Enable bit is set, the following conditions are true:

- The Tsi340 does not forward downstream any I/O transactions addressing the top 768 bytes of each aligned 1 kB block. Only those transactions addressing the bottom 256 bytes of an aligned 1 kB block inside the base and limit I/O address range are forwarded downstream. Transactions above the 64 kB I/O address boundary are forwarded as defined by the address range defined by the I/O base and limit registers.
- The Tsi340 forwards upstream those I/O transactions addressing the top 768 bytes of each aligned 1 kB block within the first 64 kB of I/O space. The master enable bit in the command configuration register must also be set to enable upstream forwarding (see “Primary Command Register—Offset 0x04” on page 108). All other I/O transactions initiated on the secondary bus are forwarded upstream only if they fall outside the I/O address range.
- When the ISA enable bit is set, devices downstream of Tsi340 can have I/O space mapped into the first 256 bytes of each 1 kB chunk below the 64 kB boundary, or anywhere in I/O space above the 64 kB boundary.

## 3.4 Memory Address Decoding

Tsi340 has three mechanisms for defining memory address ranges for forwarding of memory transactions:

- Memory-mapped I/O base and limit address registers
- Prefetchable memory base and limit address registers
- VGA mode

To enable downstream forwarding of memory transactions, the memory enable bit must be set in the command register in Tsi340 configuration space (see “Primary Command Register—Offset 0x04” on page 108). To enable upstream forwarding of memory transactions, the master enable bit must be set in the command register (see “Primary Command Register—Offset 0x04” on page 108). Setting the master enable bit also allows upstream forwarding of I/O transactions.

If any Tsi340 configuration state affecting memory transaction forwarding is changed by configuration write operation on the primary bus at the same time that memory transactions are ongoing on the secondary bus, Tsi340 response to the secondary bus memory transactions is not predictable.

Configure the memory-mapped I/O base and limit address registers, prefetchable memory base and limit address registers, and VGA enable bit before setting the memory enable and master enable bits, and change them subsequently only when the primary and secondary PCI buses are idle.

### 3.4.1 Memory-Mapped I/O Base and Limit Address Registers

Memory-mapped I/O is also referred to as non-prefetchable memory. Memory addresses that cannot automatically be prefetched but that can conditionally prefetch based on command type should be mapped into this space. Read transactions to non-prefetchable space may exhibit side effects; this space may have non-memory-like behavior. Tsi340 prefetches in this space only if the memory read line or memory read multiple commands are used; transactions using the memory read command are limited to a single data transfer.

The memory-mapped I/O base address and memory-mapped I/O limit address registers define an address range that Tsi340 uses to determine when to forward memory commands. Tsi340 forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the memory-mapped I/O address range. Tsi340 ignores memory transactions initiated on the secondary interface that fall into this address range. Any transactions that fall outside this address range are ignored on the primary interface and are forwarded upstream from the secondary interface (provided that they do not fall into the prefetchable memory range or are not forwarded downstream by the VGA mechanism).

The memory-mapped I/O range supports 32-bit addressing only. The *PCI-to-PCI Bridge Architecture Specification* does not provide for 64-bit addressing in the memory-mapped I/O space. The memory-mapped I/O address range has a granularity and alignment of 1 MB. The maximum memory-mapped I/O address range is 4 GB.

The memory-mapped I/O address range is defined by a 16-bit memory-mapped I/O base address register at configuration offset 20h and by a 16-bit memory-mapped I/O limit address register at offset 22h. The top 12 bits of each of these registers correspond to bits <31:20> of the memory address. The low 4 bits are hardwired to 0. The low 20 bits of the memory-mapped I/O base address are assumed to be 0x00000, which results in a natural alignment to a 1 MB boundary. The low 20 bits of the memory-mapped I/O limit address are assumed to be F FFFFh, which results in an alignment to the top of a 1 MB block.

#### 3.4.1.1 Turning off Memory Mapped I/O

To turn off the memory-mapped I/O address range, write the memory-mapped I/O base address register with a value greater than that of the memory-mapped I/O limit address register.

The initial state of the memory-mapped I/O base address register is 0x0000 0000.The initial state of the memory-mapped I/O limit address register is 0x000F FFFF. Note that the initial states of these registers define a memory-mapped I/O range at the bottom 1 MB block of memory. Write these registers with their appropriate values before setting either the memory enable bit or the master enable bit in the command register in configuration space.

### 3.4.2 Prefetchable Memory Base and Limit Address Registers

Locations accessed in the prefetchable memory address range must have true memory-like behavior and must not exhibit side effects when read. This means that extra reads to a prefetchable memory location must have no side effects. Tsi340 prefetches for all types of memory read commands in this address space.

The prefetchable memory base address and prefetchable memory limit address registers define an address range that Tsi340 uses to determine when to forward memory commands. Tsi340 forwards a memory transaction from the primary to the secondary interface if the transaction address falls within the prefetchable memory address range. Tsi340 ignores memory transactions initiated on the secondary interface that fall into this address range. Tsi340 does not respond to any transactions that fall outside this address range on the primary interface and forwards those transactions upstream from the secondary interface (provided that they do not fall into the memory-mapped I/O range or are not forwarded by the VGA mechanism).

The prefetchable memory range supports 64-bit addressing and provides additional registers to define the upper 32 bits of the memory address range, the prefetchable memory base address upper 32 bits register, and the prefetchable memory limit address upper 32 bits register. For address comparison, a single address cycle (32-bit address) prefetchable memory transaction is treated like a 64-bit address transaction where the upper 32 bits of the address are equal to 0. This upper 32-bit value of 0 is compared to the prefetchable memory base address upper 32 bits register and the prefetchable memory limit address upper 32 bits register. The prefetchable memory base address upper 32 bits register must be 0 in order to pass any single address cycle transactions downstream. “Prefetchable Memory 64-Bit Addressing Registers” on page 52 describes 64-bit addressing support.

The prefetchable memory address range has a granularity and alignment of 1 MB. The maximum memory address range is 4 GB when 32-bit addressing is used, and above 4 GB when 64-bit addressing is used. The prefetchable memory address range is defined by a 16-bit prefetchable memory base address register at configuration offset 24h and by a 16-bit prefetchable memory limit address register at offset 0x26. The top 12 bits of each of these registers correspond to bits <31:20> of the memory address. The low 4 bits are hardwired to 1h, indicating 64-bit address support. The low 20 bits of the prefetchable memory base address are assumed to be 0 0000h, which results in a natural alignment to a 1 MB boundary. The low 20 bits of the prefetchable memory limit address are assumed to be 0xFFFFF, which results in an alignment to the top of a 1 MB block.

The initial state of the prefetchable memory base address register is 0x0000 0000. The initial state of the prefetchable memory limit address register is 0x000F FFFF. Note that the initial states of these registers define a prefetchable memory range at the bottom 1 MB block of memory. Write these registers with their appropriate values before setting either the memory enable bit or the master enable bit in the command register in configuration space.

To turn off the prefetchable memory address range, write the prefetchable memory base address register with a value greater than that of the prefetchable memory limit address register. The entire base value must be greater than the entire limit value, meaning that the upper 32 bits must be considered. Therefore, to disable the address range, the upper 32 bits registers can both be set to the same value, while the lower base register is set greater than the lower limit register; otherwise, the upper 32-bit base must be greater than the upper 32-bit limit.

### 3.4.3 Prefetchable Memory 64-Bit Addressing Registers

Tsi340 supports 64-bit memory address decoding for forwarding of dual address memory transactions. The dual address cycle is used to support 64-bit addressing. The first address phase of a dual address transaction contains the lower 32 address bits, and the second address phase contains the upper 32 address bits. During a dual address cycle transaction, the upper 32 bits must never be 0 - use the single address cycle commands for transactions addressing the first 4 GB of memory space.

Tsi340 implements the prefetchable memory base address upper 32 bits register and the prefetchable memory limit address upper 32 bits register to define a prefetchable memory address range greater than 4 GB.

The prefetchable address space can then be defined in the following ways:

- Residing entirely in the first 4 GB of memory
- Residing entirely above the first 4 GB of memory
- Crossing the first 4 GB memory boundary

#### 3.4.3.1 Residing Entirely in the First 4 GB of Memory

If the prefetchable memory space on the secondary interface resides entirely in the first 4 GB of memory, both upper 32 bits registers must be set to 0. Tsi340 ignores all dual address cycle transactions initiated on the primary interface and forwards all dual address transactions initiated on the secondary interface upstream.

#### 3.4.3.2 Residing Entirely above the First 4 GB of Memory

If the secondary interface prefetchable memory space resides entirely above the first 4 GB of memory, both the prefetchable memory base address upper 32 bits register and the prefetchable memory limit address upper 32 bits register must be initialized to nonzero values. Tsi340 ignores all single address memory transactions initiated on the primary interface and forwards all single address memory transactions initiated on the secondary interface upstream (unless they fall within the memory-mapped I/O or VGA memory range).

A dual address memory transaction is forwarded downstream from the primary interface if it falls within the address range defined by the prefetchable memory base address, prefetchable memory base address upper 32 bits, prefetchable memory limit address, and prefetchable memory limit address upper 32 bits registers. If the dual address transaction initiated on the secondary interface falls outside this address range, it is forwarded upstream to the primary interface.

Tsi340 does not respond to a dual address transaction initiated on the primary interface that falls outside this address range, or to a dual address transaction initiated on the secondary interface that falls within the address range.

#### 3.4.3.3 Crosses the First 4 GB of Memory

If the secondary interface prefetchable memory space straddles the first 4 GB address boundary, the prefetchable memory base address upper 32 bits register is set to 0, while the prefetchable memory limit address upper 32 bits register is initialized to a nonzero value. Single address cycle memory transactions are compared to the prefetchable memory base address register only.

A transaction initiated on the primary interface is forwarded downstream if the address is greater than or equal to the base address. A transaction initiated on the secondary interface is forwarded upstream if the address is less than the base address.

Dual address transactions are compared to the prefetchable memory limit address and the prefetchable memory limit address upper 32 bits registers. If the address of the dual address transaction is less than or equal to the limit, the transaction is forwarded downstream from the primary interface and is ignored on the secondary interface. If the address of the dual address transaction is greater than this limit, the transaction is ignored on the primary interface and is forwarded upstream from the secondary interface.

The prefetchable memory base address upper 32 bits register is located at configuration Dword offset 28h, and the prefetchable memory limit address upper 32 bits register is located at configuration Dword offset 2Ch. Both registers are reset to 0.

## 3.5 VGA Support

Tsi340 provides two modes for VGA support:

- VGA mode, supporting VGA-compatible addressing
- VGA snoop mode, supporting VGA palette forwarding

### 3.5.1 VGA Mode

When a VGA-compatible device exists downstream from Tsi340, set the VGA enable bit in the “Bridge Control Register—Offset 0x3C” on page 122 to enable VGA mode. When Tsi340 is operating in VGA mode, it forwards downstream those transactions addressing the VGA frame buffer memory and VGA I/O registers, regardless of the values of Tsi340 base and limit address registers. Tsi340 ignores transactions initiated on the secondary interface addressing these locations.

The VGA frame buffer consists of the following memory address range: 0x000A 0000—0x00B FFFF.

Read transactions to frame buffer memory are treated as non-prefetchable. Tsi340 requests only a single data transfer from the target, and read byte enable bits are forwarded to the target bus.

The VGA I/O addresses consist of the following I/O addresses:

- 0x3B0 – 0x3BB
- 0x3C0 – 0x3DF

These I/O addresses are aliased every 1 kB throughout the first 64 kB of I/O space. This means that address bits <15:10> are not decoded and can be any value, while address bits <31:16> must be all zero.

VGA BIOS addresses starting at 0xC0000 are not decoded in VGA mode.

### 3.5.2 VGA Snoop Mode

Tsi340 provides VGA snoop mode, allowing for VGA palette write transactions to be forwarded downstream. This mode is used when a graphics device downstream from Tsi340 needs to snoop or respond to VGA palette write transactions. To enable the mode, set the VGA snoop bit in the command register in configuration space (see “Primary Command Register—Offset 0x04” on page 108).

Tsi340 claims VGA palette write transactions by asserting DEVSEL_b in VGA snoop mode.

When the VGA snoop bit is set, Tsi340 forwards downstream transactions with the following I/O addresses:

- 0x3C6
- 0x3C8
- 0x3C9

These addresses are also forwarded as part of the VGA compatibility mode previously described. Again, address bits <15:10> are not decoded, while address bits <31:16> must be equal to 0, which means that these addresses are aliased every 1 KB throughout the first 64 KB of I/O space.

If both the VGA mode bit and the VGA snoop bit are set, Tsi340 behaves in the same way as if only the VGA mode bit were set.

# 4. Transaction Ordering

This chapter discusses the following topics about the Tsi340:



## 4.1 Overview of Transaction Ordering

To maintain data coherency and consistency, Tsi340 complies with the ordering rules set forth in the *PCI Local Bus Specification, Revision 2.3*, for transactions crossing the bridge.

This chapter describes the ordering rules that control transaction forwarding across Tsi340. For a more detailed discussion of transaction ordering, see Appendix E of the *PCI Local Bus Specification, Revision 2.3*.

## 4.2 Transaction Governed by Ordering Rules

Ordering relationships are established for the following classes of transactions crossing Tsi340:

- Posted write transactions: comprised of memory write and memory write and invalidate transactions. Posted write transactions complete at the source before they complete at the destination; that is, data is written into intermediate data buffers before it reaches the target.
- Delayed write request transactions: comprised of I/O write and configuration write transactions. Delayed write requests are terminated by target retry on the initiator bus and are queued in the delayed transaction queue. A delayed write transaction must complete on the target bus before it completes on the initiator bus.
- Delayed write completion transactions: comprised of I/O write and configuration write transactions. Delayed write completion transactions have been completed on the target bus, and the target response is queued in Tsi340 buffers. A delayed write completion transaction proceeds in the direction opposite that of the original delayed write request; that is, a delayed write completion transaction proceeds from the target bus to the initiator bus.
- Delayed read request transactions: comprised of all memory read, I/O read, and configuration read transactions. Delayed read requests are terminated by target retry on the initiator bus and are queued in the delayed transaction queue.
- Delayed read completion transactions: comprised of all memory read, I/O read, and configuration read transactions. Delayed read completion transactions have been completed on the target bus, and the read data has been queued in Tsi340 read data buffers. A delayed read completion transaction proceeds in the direction opposite that of the original delayed read request; that is, a delayed read completion transaction proceeds from the target bus to the initiator bus.

### 4.2.1 Combining or Merging Transactions

Tsi340 does not combine or merge write transactions The following rules are used in regard to combining and merging:

- Tsi340 does not combine separate write transactions into a single write transaction—this optimization is best implemented in the originating master.
- Tsi340 does not merge bytes on separate masked write transactions to the same Dword address—this optimization is also best implemented in the originating master.
- Tsi340 does not collapse sequential write transactions to the same address into a single write transaction—the *PCI Local Bus Specification* does not permit this combining of transactions.

## 4.3 General Ordering Guidelines

Independent transactions on the primary and secondary buses have a relationship only when those transactions cross Tsi340.

The following general ordering guidelines govern transactions crossing Tsi340:

- The ordering relationship of a transaction with respect to other transactions is determined when the transaction completes, that is, when a transaction ends with a termination other than target retry.
- Requests terminated with target retry can be accepted and completed in any order with respect to other transactions that have been terminated with target retry. If the order of completion of delayed requests is important, the initiator should not start a second delayed transaction until the first one has been completed. If more than one delayed transaction is initiated, the initiator should repeat all the delayed transaction requests, using some fairness algorithm. Repeating a delayed transaction cannot be contingent on completion of another delayed transaction; otherwise, a deadlock can occur.
- Write transactions flowing in one direction have no ordering requirements with respect to write transactions flowing in the other direction. Tsi340 can accept posted write transactions on both interfaces at the same time, as well as initiate posted write transactions on both interfaces at the same time.
- The acceptance of a posted memory write transaction as a target can never be contingent on the completion of a non-locked, non-posted transaction as a master. This is true of Tsi340 and must be true of other bus agents; otherwise, a deadlock can occur.
- Tsi340 accepts posted write transactions, regardless of the state of completion of any delayed transactions being forwarded across Tsi340.

### 4.3.1 Ordering Rules

Table 9 shows the ordering relationships of all the transactions and refers by number to the ordering rules that follow.

The superscript accompanying some of the table entries refers to any applicable ordering rule listed in this section. Many entries are not governed by these ordering rules; therefore, the implementation can choose whether the transactions pass each other.

The entries without superscripts reflect Tsi340's implementation choices.

**Table 9: Summary of Transaction Ordering**

| Bus Operation | Can Row Pass Column? | | | | |
|---|---|---|---|---|---|
| | Posted Write | Delayed Read Request | Delayed Write Request | Delayed Read Completion | Delayed Write Completion |
| Posted Write | No[1] | Yes[5] | Yes[5] | Yes[5] | Yes[5] |
| Delayed Read Request | No[2] | No | No | Yes | Yes |
| Delayed Write Request | No[4] | No | No | Yes | Yes |
| Delayed Read Completion | No[3] | Yes | Yes | No | No |
| Delayed Write Completion | Yes | Yes | Yes | No | No |

The following ordering rules describe the transaction relationships. Each ordering rule is followed by an explanation. These ordering rules apply to posted write transactions, delayed write and read requests, and delayed write and read completion transactions crossing Tsi340 in the same direction. Note that delayed completion transactions cross Tsi340 in the direction opposite that of the corresponding delayed requests.

1. Posted write transactions must complete on the target bus in the order in which they were received on the initiator bus. The subsequent posted write transaction can be setting a flag that covers the data in the first posted write transaction; if the second transaction were to complete before the first transaction, a device checking the flag could subsequently consume stale data.
2. A delayed read request traveling in the same direction, as a previously queued posted write transaction must push the posted write data ahead of it. The posted write transaction must complete on the target bus before the delayed read request can be attempted on the target bus. The read transaction can be to the same location as the write data, so if the read transaction were to pass the write transaction, it would return stale data.
3. A delayed read completion must "pull" ahead of previously queued posted write data traveling in the same direction. In this case, the read data is traveling in the same direction as the write data and the initiator of the read transaction is on the same side of Tsi340 as the target of the write transaction. The posted write transaction must complete to the target before the read data is returned to the initiator. The read transaction can be to a status register of the initiator of the posted write data and therefore should not complete until the write transaction is complete.

4. Delayed write requests cannot pass previously queued posted write data. As in the case of posted memory write transactions, the delayed write transaction can be setting a flag that covers the data in the posted write transaction; if the delayed write request were to complete before the earlier posted write transaction, a device checking the flag could subsequently consume stale data.
5. Posted write transactions must be given opportunities to pass delayed read and write requests and completions. Otherwise, deadlocks may occur when bridges that support delayed transactions are used in the same system with bridges that do not support-delayed transactions. A fairness algorithm is used to arbitrate between the posted write queue and the delayed transaction queue.

# 5. PCI Bus Arbitration

This chapter discusses the following:



## 5.1 Overview of PCI Bus Arbitration

Tsi340 must arbitrate for use of the primary bus when forwarding upstream transactions, and for use of the secondary bus when forwarding downstream transactions. The arbiter for the primary bus resides external to Tsi340, typically on the motherboard. For the secondary PCI bus, Tsi340 implements an internal arbiter.

This chapter describes primary and secondary bus arbitration.

## 5.2 Primary PCI Bus Arbitration

Tsi340 implements a request output pin, P_REQ_b, and a grant input pin, P_GNT_b, for primary PCI bus arbitration. Tsi340 asserts P_REQ_b when forwarding transactions upstream; that is, it acts as initiator on the primary PCI bus. As long as at least one pending transaction resides in the queues in the upstream direction, either posted write data or delayed transaction requests, Tsi340 keeps P_REQ_b asserted. However, if a target retry, target disconnect, or a target abort is received in response to a transaction initiated by Tsi340 on the primary PCI bus, Tsi340 de-asserts P_REQ_b for two PCI clock cycles.

### 5.2.1 Transactions

When P_GNT_b is asserted low by the primary bus arbiter after Tsi340 has asserted P_REQ_b, Tsi340 initiates a transaction on the primary bus during the next PCI clock cycle. If P_GNT_b is asserted to the Tsi340 and the Tsi340's P_REQ_b is not asserted, the Tsi340 parks P_AD, P_CBE_b, and P_PAR by driving them to valid logic levels. When the primary bus is parked at Tsi340 and the Tsi340 has a transaction to initiate on the primary bus, Tsi340 starts the transaction (asserts FRAME#) if P_GNT_b was asserted during the previous cycle.

#### 5.2.1.1 Posted Writes

For posted write transactions (see "Posted Write Transactions" on page 26), P_REQ_b is asserted a few cycles after S_DEVSEL_b is asserted.

#### 5.2.1.2 Delayed Reads and Writes

For delayed read and write requests (see “Delayed Transaction” on page 43), P_REQ_b is not asserted until the transaction request has been completely queued in the delayed transaction queue (target retry has been returned to the initiator) and is at the head of the delayed transaction queue.

## 5.3 Secondary PCI Bus Arbitration

Tsi340 implements an internal secondary PCI bus arbiter. This arbiter supports four external masters in addition to Tsi340.

### 5.3.1 Secondary Bus Arbitration Using the Internal Arbiter

Tsi340 has four secondary bus request input pins, S_REQ_b<3:0>, and four secondary bus output grant pins, S_GNT_b<3:0>, to support external secondary bus masters. The Tsi340 specific secondary bus request and grant signals are connected internally to the arbiter and are not brought out to external pins when S_CFN_b is low.

Tieing S_CFN_b low on the board enables the Tsi340 internal arbiter.

#### 5.3.1.1 Arbiter Priorities

The secondary arbiter supports a programmable 2-level rotating algorithm. Two groups of masters are assigned, a high priority group and a low priority group. The low priority group as a whole represents one entry in the high priority group; that is, if the high priority group consists of n masters, then in at least every n+1 transactions the highest priority is assigned to the low priority group. Priority rotates evenly among the low priority group. Therefore, members of the high priority group can be serviced n transactions out of n+1, while one member of the low priority group is serviced once every n+1 transactions.

Each bus master, including Tsi340, can be configured to be in either the low priority group or the high priority group by setting the corresponding priority bit in the arbiter control register in device-specific configuration space. Each master has a corresponding bit. If the bit is set to 1, the master is assigned to the high priority group. If the bit is set to 0, the master is assigned to the low priority group. If all the masters are assigned to one group, the algorithm defaults to a straight rotating priority among all the masters. After reset, all external masters are assigned to the low priority group, and Tsi340 is assigned to the high priority group. Tsi340 receives highest priority on the target bus every other transaction, and priority rotates evenly among the other masters.

Priorities are re-evaluated every time S_FRAME_b is asserted, that is, at the start of each new transaction on the secondary PCI bus. From this point until the time that the next transaction starts, the arbiter asserts the grant signal corresponding to the highest priority request that is asserted. When priorities are reevaluated, the highest priority is assigned to the next highest priority master relative to the master that initiated the previous transaction. The master that initiated the last transaction now has the lowest priority in its group.

If Tsi340 detects that an initiator has failed to assert S_FRAME_b after 16 cycles of both grant assertion and a secondary idle bus condition, the arbiter de-asserts the grant.

#### 5.3.1.2 Bus Contention

To prevent bus contention, if the secondary PCI bus is idle, the arbiter never asserts one grant signal in the same PCI cycle in which it de-asserts another. It de-asserts one grant, and then asserts the next grant, no earlier than one PCI clock cycle later. If the secondary PCI bus is busy, that is, either S_FRAME_b or S_IRDY_b is asserted, the arbiter can de-assert one grant and assert another grant during the same PCI clock cycle.

## 5.4 Bus Parking

Bus parking refers to driving the AD, C/BE#, and PAR lines to a known value while the bus is idle. In general, the device implementing the bus arbiter is responsible for parking the bus or assigning another device to park the bus. A device parks the bus when the bus is idle, its bus grant is asserted, and the device's request is not asserted. The AD and C/BE# signals should be driven first, with the PAR signal driven one cycle later.

Tsi340 parks the primary bus only when P_GNT_b is asserted, P_REQ_b is de-asserted, and the primary PCI bus is idle. When P_GNT_b is de-asserted, Tsi340 tristates the P_AD, P_CBE_b, and P_PAR signals on the next PCI clock cycle. If Tsi340 is parking the primary PCI bus and wants to initiate a transaction on that bus, then Tsi340 can start the transaction on the next PCI clock cycle by asserting P_FRAME_b if P_GNT_b is still asserted.

The internal secondary bus arbiter always parks the bus at the last master that used the PCI bus. That is, Tsi340 keeps the secondary bus grant asserted to a particular master until a new secondary bus request comes along. After reset, Tsi340 parks the secondary bus at itself until transactions start occurring on the secondary bus.

# 6. Error Handling

This chapter discusses the following topics about Tsi340's error handling capabilities:



## 6.1 Overview

Tsi340 checks, forwards, and generates parity on both the primary and secondary interfaces. To maintain transparency, Tsi340 always tries to forward the existing parity condition on one bus to the other bus, along with address and data. Tsi340 always attempts to be transparent when reporting errors, but this is not always possible, given the presence of posted data and delayed transactions.

To support error reporting on the PCI bus, Tsi340 implements the following features:

- PERR# and SERR# signals on both the primary and secondary interfaces (see "Signals and Pinout" on page 79)
- Primary status register and secondary status register (see "Primary Status Register—Offset 0x04" on page 110 and "Secondary Status Register—Offset 0x1C" on page 116)
- The device-specific P_SERR_b event disable register (see "P_SERR_b Event Disable Register—Offset 0x64" on page 130)
- The device-specific P_SERR_b status register (see "P_SERR_b Status Register — Offset 0x68" on page 132)

This chapter provides detailed information about how Tsi340 handles errors. It also describes error status reporting and error operation disabling.

## 6.2 Address Parity Errors

Tsi340 checks address parity for all transactions on both buses, for all address and all bus commands.

When Tsi340 detects an address parity error on the primary interface, the following events occur:

- If the parity error response bit is set in the command register (see “Primary Command Register—Offset 0x04” on page 108), Tsi340 does not claim the transaction with P_DEVSEL_b; this may allow the transaction to terminate in a master abort. If the parity error response bit is not set, Tsi340 proceeds normally and accepts the transaction if it is directed to or across Tsi340.
- Tsi340 sets the detected parity error bit in the status register (see “Primary Status Register—Offset 0x04” on page 110).
- Tsi340 asserts P_SERR_b and sets the signaled system error bit in the status register, if both of the following conditions are met:
  - The SERR# enable bit is set in the command register.
  - The parity error response bit is set in the command register

When Tsi340 detects an address parity error on the secondary interface, the following events occur:

- If the parity error response bit is set in the bridge control register (see “Bridge Control Register—Offset 0x3C” on page 122), Tsi340 does not claim the transaction with S_DEVSEL_b; this may allow the transaction to terminate in a master abort. If the parity error response bit is not set, Tsi340 proceeds normally and accepts the transaction if it is directed to or across Tsi340.
- Tsi340 sets the detected parity error bit in the secondary status register.
- Tsi340 asserts P_SERR_b and sets the signaled system error bit in the status register (see “Secondary Status Register—Offset 0x1C” on page 116), if both of the following conditions are met:
  - The SERR# enable bit is set in the command register.
  - The parity error response bit is set in the bridge control register (see “Bridge Control Register—Offset 0x3C” on page 122).

## 6.3 Data Parity Errors

When forwarding transactions, Tsi340 attempts to pass the data parity condition from one interface to the other unchanged, whenever possible, to allow the master and target devices to handle the error condition.

The following sections describe, for each type of transaction, the sequence of events that occurs when a parity error is detected and the way in which the parity condition is forwarded across Tsi340.

### 6.3.1 Configuration Write Transactions to Tsi340 Configuration Space

When Tsi340 detects a data parity error during a Type 0 configuration write transaction to Tsi340 configuration space, the following events occur:

- If the parity error response bit is set in the command register, Tsi340 asserts P_TRDY_b and writes the data to the configuration register. Tsi340 also asserts P_PERR_b.
- If the parity error response bit is not set, Tsi340 does not assert P_PERR_b.
- Tsi340 sets the detected parity error bit in the status register, regardless of the state of the parity error response bit.

### 6.3.2 Read Transactions

When Tsi340 detects a parity error during a read transaction, the target drives data and data parity, and the initiator checks parity and conditionally asserts PERR#.

#### 6.3.2.1 Downstream Transaction

For downstream transactions, when Tsi340 detects a read data parity error on the secondary bus, the following events occur:

- Tsi340 asserts S_PERR_b two cycles following the data transfer, if the secondary interface parity error response bit is set in the bridge control register (see “Bridge Control Register—Offset 0x3C” on page 122).
- Tsi340 sets the detected parity error bit in the secondary status register (see “Secondary Status Register—Offset 0x1C” on page 116).
- Tsi340 sets the data parity detected bit in the secondary status register, if the secondary interface parity error response bit is set in the bridge control register.
- Tsi340 forwards the bad parity with the data back to the initiator on the primary bus.

##### *Prefetched Reads*

If the data with the bad parity is prefetched and is not read by the initiator on the primary bus, the data is discarded and the data with bad parity is not returned to the initiator. Tsi340 completes the transaction normally.

#### 6.3.2.2 Upstream Transaction

For upstream transactions, when Tsi340 detects a read data parity error on the primary bus, the following events occur:

- Tsi340 asserts P_PERR_b two cycles following the data transfer, if the primary interface parity error response bit is set in the command register
- Tsi340 sets the detected parity error bit in the primary status register.
- Tsi340 sets the data parity detected bit in the primary status register, if the primary interface parity error response bit is set in the command register.
- Tsi340 forwards the bad parity with the data back to the initiator on the secondary bus.

##### *Prefetched Read*

If the data with the bad parity is prefetched and is not read by the initiator on the secondary bus, the data is discarded and the data with bad parity is not returned to the initiator. Tsi340 completes the transaction normally.

#### 6.3.2.3 PERR#

Tsi340 returns to the initiator the data and parity that was received from the target. When the initiator detects a parity error on this read data and is enabled to report it, the initiator asserts PERR# two cycles after the data transfer occurs. It is assumed that the initiator takes responsibility for handling a parity error condition; therefore, when Tsi340 detects PERR# asserted while returning read data to the initiator, Tsi340 does not take any further action and completes the transaction normally.

### 6.3.3 Delayed Write Transactions

When Tsi340 detects a data parity error during a delayed write transaction, the initiator drives data and data parity, and the target checks parity and conditionally asserts PERR#.

For delayed write transactions, a parity error can occur at the following times:

- During the original delayed write request transaction
- When the initiator repeats the delayed write request transaction
- When Tsi340 completes the delayed write transaction to the target

When a delayed write transaction is normally queued, the address, command, address parity, data, byte enable bits, and data parity are all captured and a target retry is returned to the initiator.

#### 6.3.3.1 Parity Error on Initial Delayed Write Request

When Tsi340 detects a parity error on the write data for the initial delayed write request transaction, the following events occur:

- If the parity error response bit corresponding to the initiator bus is set, Tsi340 asserts TRDY# to the initiator and the transaction is not queued. If multiple data phases are requested, STOP# is also asserted to cause a target disconnect. Two cycles after the data transfer, Tsi340 also asserts PERR#. If the parity error response bit is not set, Tsi340 returns a target retry and queues the transaction as usual. Signal PERR# is not asserted. In this case, the initiator repeats the transaction.
- Tsi340 sets the detected parity error bit in the status register corresponding to the initiator bus, regardless of the state of the parity error response bit.

If parity checking is turned off and data parity errors have occurred for queued or subsequent delayed write transactions on the initiator bus, it is possible that the initiator's reattempts of the write transaction may not match the original queued delayed write information contained in the delayed transaction queue. In this case, a master time-out condition may occur, possibly resulting in a system error (P_SERR_b asserted).

##### *Downstream Transactions*

For downstream transactions, when Tsi340 is delivering data to the target on the secondary bus and S_PERR_b is asserted by the target, the following events occur:

- Tsi340 sets the secondary interface data parity detected bit in the secondary status register, if the secondary parity error response bit is set in the bridge control register.
- Tsi340 captures the parity error condition to forward it back to the initiator on the primary bus.

##### *Upstream Transactions*

For upstream transactions, when Tsi340 is delivering data to the target on the primary bus and P_PERR_b is asserted by the target, the following events occur:

- Tsi340 sets the primary interface data parity detected bit in the status register, if the primary parity error response bit is set in the command register.
- Tsi340 captures the parity error condition to forward it back to the initiator on the secondary bus.

#### 6.3.3.2 Parity Error on Delayed Write Request on a Reattempt or Forwarded Transaction

A delayed write transaction is completed on the initiator bus when the initiator repeats the write transaction with the same address, command, data, and byte enable bits as the delayed write command that is at the head of the posted data queue. Note that the parity bit is not compared when determining whether the transaction matches those in the delayed transaction queues.

Two cases must be considered:

- When parity error is detected on the initiator bus on a subsequent reattempt of the transaction and was not detected on the target bus.
- When parity error is forwarded back from the target bus.

##### *Downstream Transactions*

For downstream delayed write transactions, when the parity error is detected on the initiator bus and Tsi340 has write status to return, the following events occur:

- Tsi340 first asserts P_TRDY_b and then asserts P_PERR_b two cycles later, if the primary interface parity error response bit is set in the command register.
- Tsi340 sets the primary interface parity error detected bit in the status register.
- Because there was not an exact data and parity match, the write status is not returned and the transaction remains in the queue.

##### *Upstream Transactions*

Similarly, for upstream delayed write transactions, when the parity error is detected on the initiator bus and Tsi340 has write status to return, the following events occur:

- Tsi340 first asserts S_TRDY_b and then asserts S_PERR_b two cycles later, if the secondary interface parity error response bit is set in the bridge control register.
- Tsi340 sets the secondary interface parity error detected bit in the secondary status register.
- Because there was not an exact data and parity match, the write status is not returned and the transaction remains in the queue.

##### *Parity Error on a Forwarded Transaction from the Target Bus to the Initiator Bus*

For downstream transactions, in the case where the parity error is being passed back from the target bus and the parity error condition was not originally detected on the initiator bus, the following events occur:

- Tsi340 asserts P_PERR_b two cycles after the data transfer, if both of the following are true:
  - The primary interface parity error response bit is set in the command register.
  - The secondary interface parity error response bit is set in the bridge control register.
- Tsi340 completes the transaction normally.

For upstream transactions, in the case where the parity error is being passed back from the target bus and the initiator bus, the following events occur:

- Tsi340 asserts S_PERR_b two cycles after the data transfer, if both of the following are true:

— The primary interface parity error response bit is set in the command register.

— The secondary interface parity error response bit is set in the bridge control register.

- Tsi340 completes the transaction normally.

### 6.3.4 Posted Write Transactions

The following sections detail different types of posted write transactions and error conditions.

#### 6.3.4.1 Data Parity Error on the Initiator (Primary) Bus

When the Tsi340, responding as a target on a downstream-posted write transactions, detects a data parity error on the initiator (primary) bus, the following events occur:

- Tsi340 asserts P_PERR_b two cycles after the data transfer, if the primary interface parity error response bit is set in the command register.
- Tsi340 sets the primary interface parity error detected bit in the status register.
- Tsi340 captures and forwards the bad parity condition to the secondary bus.
- Tsi340 completes the transaction normally.

Similarly, during upstream posted write transactions, when Tsi340, responding as a target, detects a data parity error on the initiator (secondary) bus, the following events occur:

- Tsi340 asserts S_PERR_b two cycles after the data transfer, if the secondary interface parity error response bit is set in the bridge control register.
- Tsi340 sets the secondary interface parity error detected bit in the secondary status register.
- Tsi340 captures and forwards the bad parity condition to the primary bus.
- Tsi340 completes the transaction normally.

#### 6.3.4.2 Data Parity Error Reported on the Target (Secondary) Bus

During downstream write transactions, when a data parity error is reported on the target (secondary) bus by the target's assertion of S_PERR_b, the following events occur:

- Tsi340 sets the data parity detected bit in the secondary status register, if the secondary interface parity error response bit is set in the bridge control register.
- Tsi340 asserts P_SERR_b and sets the signaled system error bit in the status register, if all of the following conditions are met:
  — The SERR# enable bit is set in the command register.
  — The device-specific P_SERR_b disable bit for posted write parity errors is not set.
  — The secondary interface parity error response bit is set in the bridge control register.
  — The primary interface parity error response bit is set in the command register.
  — Tsi340 did not detect the parity error on the primary (initiator) bus; that is, the parity error was not forwarded from the primary bus.

#### 6.3.4.3 Data Parity Error Reported on the Target (Primary) Bus

During upstream write transactions, when a data parity error is reported on the target (primary) bus by the target's assertion of P_PERR_b, the following events occur:

- Tsi340 sets the data parity detected bit in the status register, if the primary interface parity error response bit is set in the command register.
- Tsi340 asserts P_SERR_b and sets the signaled system error bit in the status register, if all of the following conditions are met:
  - The SERR# enable bit is set in the command register.
  - The secondary interface parity error response bit is set in the bridge control register.
  - The primary interface parity error response bit is set in the command register.
  - Tsi340 did not detect the parity error on the secondary (initiator) bus; that is, the parity error was not forwarded from the secondary bus.

#### 6.3.4.4 Assertion of P_SERR_b

The assertion of P_SERR_b is used to signal the parity error condition when the initiator does not know that the error occurred. Because the data has already been delivered with no errors, there is no other way to signal this information back to the initiator.

If the parity error is forwarded from the initiating bus to the target bus, P_SERR_b is not asserted.

## 6.4 System Error (SERR#) Reporting

Tsi340 uses the P_SERR_b signal to report a number of system error conditions in addition to the special case parity error conditions described in “Data Parity Errors” on page 65.

In order for the Tsi340 to assert P_SERR_b, the following must be met:

- For Tsi340 to assert P_SERR_b for any reason, the SERR# enable bit must be set in the command register.
- Whenever Tsi340 asserts P_SERR_b, Tsi340 must also set the signaled system error bit in the status register.

### 6.4.1 Assertion of P_SERR_b

In compliance with the *PCI-to-PCI Bridge Architecture Specification*, Tsi340 asserts P_SERR_b in the following situations:

- The secondary SERR# input, S_SERR_b, is asserted
- The SERR# forward enable bit is set in the bridge control register (see “Bridge Control Register—Offset 0x3C” on page 122).

When the Tsi340 assets P_SERR_b it also sets the received system error bit in the secondary status register.

Tsi340 may also assert P_SERR_b for any of the following reasons:

- Target abort detected during a posted write transaction
- Master abort detected during a posted write transaction
- Posted write data discarded after $2^{24}$ attempts to deliver ($2^{24}$ target retries received)
- Parity error reported on target bus during posted write transaction (see “Posted Write Transactions” on page 69)
- Delayed write data discarded after $2^{24}$ attempts to deliver ($2^{24}$ target retries received)
- Delayed read data cannot be transferred from target after $2^{24}$ attempts ($2^{24}$ target retries received)
- Master timeout on delayed transaction

### 6.4.2 Device Specific Reporting

The Tsi340 device-specific P_SERR_b status register reports the reason for Tsi340’s assertion of P_SERR_b (see “P_SERR_b Status Register — Offset 0x68” on page 132).

Most of the events listed in “Assertion of P_SERR_b” on page 71 have additional device-specific disable bits in the P_SERR_b event disable register (see “P_SERR_b Event Disable Register—Offset 0x64” on page 130) that make it possible to mask out P_SERR_b assertion for specific events. However, in the case of he master time-out condition, it has a SERR# enable bit for in the bridge control register and therefore does not have a device-specific disable bit.

# 7. PCI Power Management

This chapter discusses the following topics about the Tsi340:



## 7.1 PCI Power Management

Tsi340 incorporates functionality that meets the requirements of the *PCI Power Management Specification, Revision 1.1*. These features include:

- PCI Power Management registers using the Enhanced Capabilities Port (ECP) address mechanism (see "Power Management Capabilities Register—Offset 0x80" on page 136 and "Power Management Data Register—Offset 0x84" on page 137)
- Support for D0, D3hot and D3cold power management states
- Support for D0, D1, D2, D3hot, and D3cold power management states for devices behind the bridge
- Support of the B2 secondary bus power state when in the D3hot power management state (see Table 10)

Table 10 shows the states and related actions that Tsi340 performs during power management transitions. (No other transactions are permitted).

**Table 10: Power Management Transitions**

| Current State | Next State | Action |
|---|---|---|
| D0 | D3cold | Power has been removed from Tsi340.<br>A power-up reset must be performed to bring Tsi340 to D0. |
| D0 | D3hot | Unimplemented power state. In Tsi340, BPCCE pin is not available (the pin disables the secondary clocks and drives them low) |
| D0 | D2 | Unimplemented power state.<br>Tsi340 ignores the write to the power state bits (power state remains at D0). |
| D0 | D1 | Unimplemented power state.<br>Tsi340 ignore the write to the power state bits (power state remains at D0). |
| D3hot | D0 | Tsi340 enables secondary clock outputs and performs an internal chip reset. Signal S_RST_b is not asserted. All registers are returned to the reset values and buffers are cleared. |

**Table 10: Power Management Transitions**

| Current State | Next State | Action |
|---|---|---|
| D3hot | D3cold | Power has been removed from Tsi340.<br>A power-up reset must be performed to bring Tsi340 to D0. |
| D3cold | D0 | Power-up reset.<br>Tsi340 performs the standard power-up reset functions |

PME# signals are routed from downstream devices *around* PCI-to-PCI bridges. PME# signals do not pass through PCI-to-PCI bridges.

# 8. Reset, Clock, and Initialization

This chapter discusses the following topics about the Tsi340:



## 8.1 Clocking

The Tsi340 has a primary and secondary clock. The Tsi340 is a synchronous device in the sense that the Secondary Clock Outputs are synchronous to the Primary Clock input.

### 8.1.1 Primary Input

The Primary Clock Input (P_CLK) drives the primary interface. P_CLK can operate between 0 MHz and 66 MHz.

### 8.1.2 Secondary Clock Outputs

Tsi340 has four secondary clock outputs, S_CLK_0[3:0], that can be used as clock inputs for up to four external secondary bus devices.

The S_CLK_0 outputs are derived from P_CLK (that is, they are synchronous to the primary clock). When both Tsi340 PCI interfaces operate at 66MHz, the Tsi340 secondary clock outputs are identical in phase to the primary clock input (P_CLK).

The following rules should be followed regarding the secondary output clocks:

- Each secondary clock output is limited to one load.
- Unused secondary clocks should be disabled through software by writing to the "Secondary Clock Control Register—Offset 0x68" on page 131.

### 8.1.3 Clock Run

The Tsi340 supports the PCI clock run protocol defined in the *PCI Mobile Design Guide 1.0.* The P_CLKRUN_b signal is high when the system's central resource initiates to stop the primary clock (P_CLK). The Tsi340 then signals that it allows the PCI clock to be stopped by keeping P_CLKRUN# high, or it will initiate P_CLK to remain running by driving P_CLKRUN_b low for two clocks. After the two clocks have elapsed, the system's central resource will keep P_CLKRUN_b low.

There are three conditions where the bridge will keep the primary clock running:

- Bit [26] offset 6Ch is set to 1
- There is a pending transaction running through the bridge
- A secondary device requires the clock

The secondary clock run protocol is enabled by setting the Secondary CLKRUN Enable bit in the “CLKRUN Register — Offset 0x6C” on page 133. The primary is responsible for the initiation of stopping or slowing down the secondary clock. However, if the CLKRUN Mode bit is set to 1, the secondary clock is stopped when the bus is idle and there are no other cycles from the primary bus.

## 8.2 Reset

Tsi340 can either be reset by hardware by asserting the P_RST_b signal or software by programming the Chip Reset bit in the “Chip Control Register/Diagnostic Control — Offset 0x44” on page 126.

### 8.2.1 Primary Interface Reset

Tsi340 has one reset input, P_RST_b. When P_RST_b is asserted, the following events occur:

- Tsi340 immediately tristates all primary and secondary PCI interface signals.
- Tsi340 performs a chip reset.
- Registers that have default values are reset.

The P_RST_b asserting and de-asserting edges can be asynchronous to P_CLK and S_CLK.

### 8.2.2 Secondary Interface Reset

Tsi340 is responsible for driving the secondary bus reset signal, S_RST_b. Tsi340 asserts S_RST_b when any of the following conditions is met:

- P_RST_b is asserted: S_RST_b is asserted when P_RST_b is asserted on the primary interface of Tsi340.
- Programming Secondary Bus Reset, in the Bridge Control register: Software can force S_RST_b on the secondary bus by programming this bit to 1. The software must clear this bit to 0 to de-assert S_RST_b. Following the clear operation, the Tsi340 de-asserts S_RST_b after a 100 us period.
- Chip Reset in the Chip Control/Diagnostic register: Software can program this bit to 1 to reset assert both P_RST_b and S_RST_b on the secondary interface. The Tsi340 de-asserts S_RST_b automatically after 100 us and clears the bit to 0, provided the secondary reset bit is cleared in the “Bridge Control Register—Offset 0x3C” on page 122. Signal S_RST_b remains asserted until a configuration write operation clears the secondary reset bit.

Writing a 1 to the Chip Reset bit sets the Secondary Bus Reset bit in the Bridge Control register.

#### 8.2.2.1 S_RST_b Impact

When S_RST_b is asserted by means of the secondary reset bit in the Bridge Control register, the Tsi340 remains accessible during secondary interface reset and continues to respond to accesses to its configuration space from the primary interface.

When S_RST_b is asserted, all secondary PCI interface control signals, including the secondary grant outputs, are immediately tristated. Signals S_AD, S_CBE_b, and S_PAR are driven low for the duration of S_RST_b assertion. All posted write and delayed transaction data buffers are reset; therefore, any transactions residing in Tsi340 buffers at the time of secondary reset are discarded.

### 8.2.3 Chip Reset

The chip reset bit in the Chip Control/Diagnostic register can be used to reset Tsi340 and the secondary bus. During chip reset, Tsi340 is inaccessible.

When the chip reset bit is set, the entire Tsi340 is reset and all signals are tristated. In addition, S_RST_b is asserted, and the secondary reset bit is automatically set. The S_RST_b signal remains asserted until a configuration write operation clears the secondary reset bit.

As soon as chip reset completes, within 20 PCI clock cycles after completion of the configuration write operation that sets the chip reset bit, the chip reset bit automatically clears and the chip is ready for configuration.

# 9. Signals and Pinout

This chapter discusses the following topics:



## 9.1 Overview

This chapter provides detailed descriptions of Tsi340 signal pins and pinout. The signal pins are grouped by function.

**Table 11: Signal Types**

| Signal Type | Description |
|---|---|
| I | Standard input only |
| O | Standard output only |
| TS | Tristate bidirectional |
| STS | Sustained tristate. Active low signal must be pulled high for one cycle when deasserting. |
| OD | Standard open drain |

The _b signal name suffix indicates that the signal is asserted when it is at a low voltage level and corresponds to the # suffix in the *PCI Local Bus Specification*. If this suffix is not present, the signal is asserted when it is at a high voltage level.

# 9.2 Signals

## 9.2.1 Primary Bus Interface Signals

The Primary PCI Interface signals are described in the following table.

**Table 12: Primary PCI Interface Signals**

| Signal Name | Pin Number | Pin Type | Description |
|---|---|---|---|
| P_AD[31:0] | 121, 122, 123, 124, 125, 126, 127, 2, 5, 6, 7, 8, 9, 10, 12, 13, 25, 26, 27, 28, 30, 31, 32, 33, 35, 36, 37, 40, 41, 42, 43, 44 | TS | Primary PCI interface address/data.<br>These signals are a multiplexed address and data bus. During the address phase or phases of a transaction, the initiator drives a physical address on P_AD[31:0]. During the data phases of a transaction, the initiator drives write data, or the target drives read data, on P_AD[31:0]. When the primary PCI bus is idle, Tsi340 drives P_AD to a valid logic level when P_GNT_b is asserted. |
| P_CBE_b[3:0] | 3, 14, 24, 34 | TS | Transaction type on P_CBE_b[3:0].<br>When there are two address phases, the first address phase carries the dual address command and the second address phase carries the transaction type. For both read and write transactions, the initiator drives byte enables on P_CBE_b[3:0] during the data phases. When the primary PCI bus is idle, Tsi340 drives P_CBE_b to a valid logic level when P_GNT_b is asserted. |
| P_PAR | 23 | TS | Primary PCI interface parity.<br>Signal P_PAR carries the even parity of the 36 bits of P_AD[31:0] and P_CBE_b[3:0] for both address and data phases. Signal P_PAR is driven by the same agent that has driven the address (for address parity) or the data (for data parity). Signal P_PAR contains valid parity one cycle after the address is valid (indicated by assertion of P_FRAME_b, or one cycle after data is valid (indicated by assertion of P_IRDY_b for write transactions and P_TRDY_b for read transactions). Signal P_PAR is driven by the device-driving read or write data one cycle after P_AD is driven. Signal P_PAR is tristated one cycle after the P_AD lines are tristated. Devices receiving data sample P_PAR as an input to check for possible parity errors. When the primary PCI bus is idle, Tsi340 drives P_PAR to a valid logic level when P_GNT_b is asserted (one cycle after the P_AD bus is parked). |
| P_FRAME_b | 15 | STS | Primary PCI interface FRAME#.<br>Signal P_FRAME_b is driven by the initiator of a transaction to indicate the beginning and duration of an access on the primary PCI bus. Signal P_FRAME_b assertion (falling edge) indicates the beginning of a PCI transaction. While P_FRAME_b remains asserted, data transfers can continue. The deassertion of P_FRAME_b indicates the final data phase requested by the initiator. When the primary PCI bus is idle, P_FRAME_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |

**Table 12: Primary PCI Interface Signals**

| Signal Name | Pin Number | Pin Type | Description |
|---|---|---|---|
| P_IRDY_b | 16 | STS | Primary PCI interface IRDY#.<br>Signal P_IRDY_b is driven by the initiator of a transaction to indicate the initiator's ability to complete the current data phase on the primary PCI bus. During a write transaction, assertion of P_IRDY_b indicates that valid write data is being driven on the P_AD bus. During a read transaction, assertion of P_IRDY_b indicates that the initiator is able to accept read data for the current data phase. Once asserted during a given data phase, P_IRDY_b is not deasserted until the data phase completes. When the primary bus is idle, P_IRDY_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| P_TRDY_b | 17 | STS | Primary PCI interface TRDY#.<br>Signal P_TRDY_b is driven by the target of a transaction to indicate the target's ability to complete the current data phase on the primary PCI bus. During a write transaction, assertion of P_TRDY_b indicates that the target is able to accept write data for the current data phase. During a read transaction, assertion of P_TRDY_b indicates that the target is driving valid read data on the P_AD bus. Once asserted during a given data phase, P_TRDY_b is not deasserted until the data phase completes. When the primary bus is idle, P_TRDY_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| P_DEVSEL_b | 18 | STS | Primary PCI interface DEVSEL#.<br>P_DEVSEL_b is asserted by the target, indicating that the device is accepting the transaction. As a target, Tsi340 performs positive decoding on the address of a transaction initiated on the primary bus to determine whether to assert P_DEVSEL_b. As an initiator of a transaction on the primary bus, Tsi340 looks for the assertion of P_DEVSEL_b within five cycles of P_FRAME_b assertion; otherwise, Tsi340 terminates the transaction with a master abort. When the primary bus is idle, P_DEVSEL_b is<br>driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| P_STOP_b | 19 | STS | Primary PCI interface STOP#.<br>Signal P_STOP_b is driven by the target of the current transaction, indicating that the target is requesting the initiator to stop the current transaction on the primary bus.<br>• When P_STOP_b is asserted in conjunction with P_TRDY_b and P_DEVSEL_b assertion, a disconnect with data transfer is being signaled.<br>• When P_STOP_b and P_DEVSEL_b are asserted, but P_TRDY_b is deasserted, a target disconnect without data transfer is being signaled. When this occurs on the first data phase, that is, no data is transferred during the transaction, this is referred to as a target retry.<br>• When P_STOP_b is asserted and P_DEVSEL_b is deasserted, the target is signaling a target abort. When the primary bus is idle, P_STOP_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |

**Table 12: Primary PCI Interface Signals**

| Signal Name | Pin Number | Pin Type | Description |
|---|---|---|---|
| P_IDSEL | 4 | I | Primary PCI interface IDSEL#.<br>Signal P_IDSEL is used as the chip select line for Type 0 configuration accesses to Tsi340 configuration space. When P_IDSEL is asserted during the address phase of a Type 0 configuration transaction, the Tsi340 responds to the transaction by asserting P_DEVSEL_b. |
| P_PERR_b | 21 | STS | Primary PCI interface PERR#.<br>Signal P_PERR_b is asserted when a data parity error is detected for data received on the primary interface. The timing of P_PERR_b corresponds to P_PAR driven one cycle earlier and P_AD and P_CBE_b driven two cycles earlier. Signal P_PERR_b is asserted by the target during write transactions, and by the initiator during read transactions. When the primary bus is idle, P_PERR_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| P_SERR_b | 22 | OD | Primary PCI interface SERR#.<br>Signal P_SERR_b can be driven low by any device on the primary bus to indicate a system error condition. Tsi340 can assert P_SERR_b for the following reasons:<br>• Address parity error<br>• Posted write data parity error on target bus<br>• Secondary bus S_SERR_b assertion<br>• Master abort during posted write transaction<br>• Target abort during posted write transaction<br>• Posted write transaction discarded<br>• Delayed write request discarded<br>• Delayed read request discarded<br>• Delayed transaction master timeout<br>Signal P_SERR_b is pulled up through an external resistor. |
| P_REQ_b | 119 | TS | Primary PCI bus REQ#.<br>Signal P_REQ_b is asserted by the Tsi340 to indicate to the primary bus arbiter that it wants to start a transaction on the primary bus. When Tsi340 receives a target retry or disconnect in response to initiating a transaction, Tsi340 deasserts P_REQ_b for at least two PCI clock cycles before asserting it again. |
| P_GNT_b | 118 | I | Primary PCI bus GNT#.<br>When asserted, P_GNT_b indicates to Tsi340 that access to the primary bus is granted. The Tsi340 can start a transaction on the primary bus when the bus is idle and P_GNT_b is asserted. When Tsi340 has not requested use of the bus and P_GNT_b is asserted, Tsi340 must drive P_AD, and P_PAR to valid logic levels. |

### 9.2.2 Secondary Bus Interface Signals

The Secondary PCI Interface signals are described in the following table.

**Table 13: Secondary PCI Interface Signals**

| Signal Name | Pin Number | Type | Description |
|---|---|---|---|
| S_AD[31:0] | 95, 94, 92, 91, 90, 89, 88, 87, 85, 83, 82, 81, 80, 79, 78, 77, 63, 62, 61, 60, 59, 57, 56, 55, 53, 52, 51, 50, 48, 47, 46, 45 | TS | Secondary PCI interface address/data.<br>These signals are a multiplexed address and data bus. During the address phase or phases of a transaction, the initiator drives a physical address on S_AD[31:0]. During the data phases of a transaction, the initiator drives write data, or the target drives read data, on S_AD[31:0]. When the secondary PCI bus is idle, Tsi340 drives S_AD to a valid logic level when its secondary bus grant is asserted. |
| S_CBE_b[3:0] | 86, 76, 66, 54 | TS | Secondary PCI interface command/byte enables.<br>These signals are a multiplexed command field and byte enable field. During the address phase or phases of a transaction, the initiator drives the transaction type on S_CBE_b[3:0]. When there are two address phases, the first address phase carries the dual address command and the second address phase carries the transaction type. For both read and write transactions, the initiator drives byte enables on S_CBE_b[3:0] during the data phases. When the secondary PCI bus is idle, Tsi340 drives S_CBE_b to a valid logic level when its secondary bus grant is asserted. S_PAR TS Secondary PCI interface parity. Signal S_PAR. |
| S_PAR | 67 | TS | Secondary PCI interface parity.<br>Signal S_PAR carries the even parity of the 36 bits of S_AD[31:0] and S_CBE_b[3:0] for both address and data phases. Signal S_PAR is driven by the same agent that has driven the address (for address parity) or the data (for data parity). Signal S_PAR contains valid parity one cycle after the address is valid (indicated by assertion of S_FRAME_b), or one cycle after data is valid (indicated by assertion of S_IRDY_b for write transactions and S_TRDY_b for read transactions). Signal S_PAR is driven by the device driving read or write data one cycle after S_AD is driven. Signal S_PAR is tristated one cycle after the S_AD lines are tristated. Devices receiving data sample S_PAR as an input in order to check for possible parity errors. When the secondary PCI bus is idle, the Tsi340 drives S_PAR to a valid logic level when its secondary bus grant is asserted (one cycle after the S_AD bus is parked). |
| S_FRAME_b | 74 | STS | Secondary PCI interface FRAME#.<br>Signal S_FRAME_b is driven by the initiator of a transaction to indicate the beginning and duration of an access on the secondary PCI bus. Signal S_FRAME_b assertion (falling edge) indicates the beginning of a PCI transaction. While S_FRAME_b remains asserted, data transfers can continue. The deassertion of S_FRAME_b indicates the final data phase requested by the initiator. When the secondary PCI bus is idle, S_FRAME_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |

**Table 13: Secondary PCI Interface Signals**

| Signal Name | Pin Number | Type | Description |
|---|---|---|---|
| S_IRDY_b | 73 | STS | Secondary PCI interface IRDY#.<br>Signal S_IRDY_b is driven by the initiator of a transaction to indicate the initiator's ability to complete the current data phase on the secondary PCI bus. During a write transaction, assertion of S_IRDY_b indicates that valid write data is being driven on the S_AD bus. During a read transaction, assertion of S_IRDY_b indicates that the initiator is able to accept read data for the current data phase. Once asserted during a given data phase, S_IRDY_b is not deasserted until the data phase completes. When the secondary bus is idle, S_IRDY_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| S_TRDY_b | 72 | STS | Secondary PCI interface TRDY.<br>Signal S_TRDY_b is driven by the target of a transaction to indicate the target's ability to complete the current data phase on the secondary PCI bus. During a write transaction, assertion of S_TRDY_b indicates that the target is able to accept write data for the current data phase. During a read transaction, assertion of S_TRDY_b indicates that the target is driving valid read data on the S_AD bus. Once asserted during a given data phase, S_TRDY_b is not deasserted until the data phase completes. When the secondary bus is idle, S_TRDY_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| S_DEVSEL_b | 71 | STS | Secondary PCI interface DEVSEL#.<br>Signal S_DEVSEL_b is asserted by the target, indicating that the device is accepting the transaction. As a target, Tsi340 performs positive decoding on the address of a transaction initiated on the secondary bus in order to determine whether to assert S_DEVSEL_b. As an initiator of a transaction on the secondary bus, Tsi340 looks for the assertion of S_DEVSEL_b within five cycles of S_FRAME_b assertion; otherwise, Tsi340 terminates the transaction with a master abort. When the secondary bus is idle, S_DEVSEL_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| S_STOP_b | 70 | STS | Secondary PCI interface STOP#.<br>Signal S_STOP_b is driven by the target of the current transaction, indicating that the target is requesting the initiator to stop the current transaction on the secondary bus.<br>• When S_STOP_b is asserted in conjunction with S_TRDY_b and S_DEVSEL_b assertion, a disconnect with data transfer is being signaled.<br>• When S_STOP_b and S_DEVSEL_b are asserted, but S_TRDY_b is deasserted, a target disconnect without data transfer is being signaled. When this occurs on the first data phase, that is, no data is transferred during the transaction, this is referred to as a target retry.<br>• When S_STOP_b is asserted and S_DEVSEL_b is deasserted, the target is signaling a target abort. When the secondary bus is idle, S_STOP_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |

**Table 13: Secondary PCI Interface Signals**

| Signal Name | Pin Number | Type | Description |
|---|---|---|---|
| S_PERR_b | 69 | STS | Secondary PCI interface PERR#.<br>Signal S_PERR_b is asserted when a data parity error is detected for data received on the secondary interface. The timing of S_PERR_b corresponds to S_PAR driven one cycle earlier and S_AD driven two cycles earlier. Signal S_PERR_b is asserted by the target during write transactions, and by the initiator during read transactions. When the secondary bus is idle, S_PERR_b is driven to a deasserted state for one cycle and then is sustained by an external pull-up resistor. |
| S_SERR_b | 68 | I | Secondary PCI interface SERR#.<br>Signal S_SERR_b can be driven low by any device except Tsi340 on the secondary bus to indicate a system error condition. Tsi340 samples S_SERR_b as an input and conditionally forwards it to the primary bus on P_SERR_b. Tsi340 does not drive S_SERR_b. Signal S_SERR_b is pulled up through an external resistor. |
| S_REQ_b[3:0] | 99, 98, 97, 96 | I | Secondary PCI interface REQ#s.<br>Tsi340 accepts four request inputs, S_REQ_b[3:0], into its secondary bus arbiter. Tsi340 request input to the arbiter is an internal signal. Each request input can be programmed to be in either a high priority rotating group or a low priority rotating group. An asserted level on an S_REQ_b pin indicates that the corresponding master wants to initiate a transaction on the secondary PCI bus. |
| S_GNT_b[3:0] | 104, 103, 101, 100 | TS | Secondary PCI interface GNT#s.<br>Tsi340 secondary bus arbiter can assert one of four secondary bus grant outputs, S_GNT_b[3:0], to indicate that an initiator can start a transaction on the secondary bus if the bus is idle. |

### 9.2.3 Clocks and Resets

The clocking and reset signals are described in the following table.

**Table 14: Clocks and Resets**

| Signal Name | Pin Number | Type | Description |
|---|---|---|---|
| P_CLK | 117 | I | Primary Clock Input<br>Provides timing for all transactions on the primary interface. |
| S_CLK_O[3:0] | 110, 109, 108, 107 | O | Secondary Clock Output<br>Provides secondary clocks phase synchronous with the P_CLK. |
| P_CLKRUN_b | 115 | TS | Primary Clock Run<br>Allows main system to stop the primary clock based on the specifications in the *PCI Mobile Design Guid*e, *Revision 1.0*. If unused, this pin should be tied to ground to signify that P_CLK is always running. |

**Table 14: Clocks and Resets**

| Signal Name | Pin Number | Type | Description |
|---|---|---|---|
| S_CLKRUN_b | 112 | TS | Secondary Clock Run<br>Allows main system to slow down or stop the secondary clock and is controlled by the primary or bit[4] offset 6Fh. If the secondary devices do not support CLKRUN, this pin should be pulled LOW by a 300 ohm resistor. |
| P_RST_b | 116 | I | Primary RESET (Active LOW)<br>When P_RST# is active, all PCI signals should be asynchronously tri-stated. |
| S_RST_b | 105 | O | Secondary RESET (Active LOW)<br>Asserted when any of the following conditions are met:<br>• Signal P_RST_b is asserted.<br>• Chip Reset bit in bridge control register in is set (see “Bridge Control Register—Offset 0x3C” on page 122)<br>When this signal is asserted, all control signals are tri-stated and zeroes are driven on S_AD, S_CBE, and S_PAR. |

### 9.2.4 Miscellaneous Signals

Miscellaneous signals are described in the following table

**Table 15: Miscellaneous Signals**

| Signal Name | Pin Number | Type | Description |
|---|---|---|---|
| HS_ENUM_b | 113 | O | Hot Swap Status Indicator<br>The output of ENUM_b indicates to the system that an insertion has occurred or that an extraction is about to occur. |
| HS_LED_OUT | 114 | I/O | Hot Swap LED<br>The output of this pin lights an LED to indicate insertion or removal ready status. This pin may also be used as a input or detect pin. Every 500us, the pin tri-states for eight primary PCI clock cycles to sample the status. |
| SCAN_TM_b | 65 | I | Full-Scan Test Mode Enable<br>Manufacturing test pin.<br>For normal operation, pull SCAN_TM_b to high. This signal has an internal pull-up. |
| SCAN_EN | 106 | I/O | Full-Scan Enable Control<br>Manufacturing test pin.<br>For normal operation, SCAN_TM_b should be pulled high and SCAN_EN becomes an output with logic 0. |

## 9.3 Pinout

**Table 16: 128 PQFP Pinlist**

| Pin Number | Signal Name |
|---|---|
| 1 | VDD |
| 2 | P_AD[24] |
| 3 | P_CBE_b[3] |
| 4 | P_IDSEL |
| 5 | P_AD[23] |
| 6 | P_AD[22] |
| 7 | P_AD[21] |
| 8 | P_AD[20] |
| 9 | P_AD[19] |
| 10 | P_AD[18] |
| 11 | VSS |
| 12 | P_AD[17] |
| 13 | P_AD[16] |
| 14 | P_CBE_b[2] |
| 15 | P_FRAME_b |
| 16 | P_IRDY_b |
| 17 | P_TRDY_b |
| 18 | P_DEVSEL_b |
| 19 | P_STOP_b |
| 20 | VDD |
| 21 | P_PERR_b |
| 22 | P_SERR_b |
| 23 | P_PAR |
| 24 | P_CBE_b[1] |
| 25 | P_AD[15] |
| 26 | P_AD[14] |

**Table 16: 128 PQFP Pinlist**

| Pin Number | Signal Name |
|---|---|
| 27 | P_AD[13] |
| 28 | P_AD[12] |
| 29 | VSS |
| 30 | P_AD[11] |
| 31 | P_AD[10] |
| 32 | P_AD[9] |
| 33 | P_AD[8] |
| 34 | P_CBE_b[0] |
| 35 | P_AD[7] |
| 36 | P_AD[6] |
| 37 | P_AD[5] |
| 38 | VSS |
| 39 | VDD |
| 40 | P_AD[4] |
| 41 | P_AD[3] |
| 42 | P_AD[2] |
| 43 | P_AD[1] |
| 44 | P_AD[0] |
| 45 | S_AD[0] |
| 46 | S_AD[1] |
| 47 | S_AD[2] |
| 48 | S_AD[3] |
| 49 | VSS |
| 50 | S_AD[4] |
| 51 | S_AD[5] |
| 52 | S_AD[6] |
| 53 | S_AD[7] |

**Table 16: 128 PQFP Pinlist**

| Pin Number | Signal Name |
|---|---|
| 54 | S_CBE_b[0] |
| 55 | S_AD[8] |
| 56 | S_AD[9] |
| 57 | S_AD[10] |
| 58 | VDD |
| 59 | S_AD[11] |
| 60 | S_AD[12] |
| 61 | S_AD[13] |
| 62 | S_AD[14] |
| 63 | S_AD[15] |
| 64 | VSS |
| 65 | SCAN_TM_b |
| 66 | S_CBE_b[1] |
| 67 | S_PAR |
| 68 | S_SERR_b |
| 69 | S_PERR_b |
| 70 | S_STOP_b |
| 71 | S_DEVSEL_b |
| 72 | S_TRDY_b |
| 73 | S_IRDY_b |
| 74 | S_FRAME_b |
| 75 | VSS |
| 76 | S_CBE_b[2] |
| 77 | S_AD[16] |
| 78 | S_AD[17] |
| 79 | S_AD[18] |
| 80 | S_AD[19] |

**Table 16: 128 PQFP Pinlist**

| Pin Number | Signal Name |
|---|---|
| 81 | S_AD[20] |
| 82 | S_AD[21] |
| 83 | S_AD[22] |
| 84 | VDD |
| 85 | S_AD[23] |
| 86 | S_CBE_b[3] |
| 87 | S_AD[24] |
| 88 | S_AD[25] |
| 89 | S_AD[26] |
| 90 | S_AD[27] |
| 91 | S_AD[28] |
| 92 | S_AD[29] |
| 93 | VSS |
| 94 | S_AD[30] |
| 95 | S_AD[31] |
| 96 | S_REQ_b[0] |
| 97 | S_REQ_b[1] |
| 98 | S_REQ_b[2] |
| 99 | S_REQ_b[3] |
| 100 | S_GNT_b[0] |
| 101 | S_GNT_b[1] |
| 102 | VDD |
| 103 | S_GNT_b[2] |
| 104 | S_GNT_b[3] |
| 105 | S_RST_b |
| 106 | SCAN_EN |
| 107 | S_CLK_O[0] |

**Table 16: 128 PQFP Pinlist**

| Pin Number | Signal Name |
|---|---|
| 108 | S_CLK_O[1] |
| 109 | S_CLK_O[2] |
| 110 | S_CLK_O[3] |
| 111 | VSS |
| 112 | S_CLKRUN_b |
| 113 | HS_ENUM_b |
| 114 | HS_LED_OUT |
| 115 | P_CLKRUN_b |
| 116 | P_RST_b |
| 117 | P_CLK |
| 118 | P_GNT_b |
| 119 | P_REQ_b |
| 120 | VDD |
| 121 | P_AD[31] |
| 122 | P_AD[30] |
| 123 | P_AD[29] |
| 124 | P_AD[28] |
| 125 | P_AD[27] |
| 126 | P_AD[26] |
| 127 | P_AD[25] |
| 128 | VSS |

# 10. Electrical Characteristics

Topics discussed include the following:



## 10.1 Absolute Maximum Ratings

Table 17 contains the absolute maximum ratings for the Tsi340.

**Table 17: Absolute Maximum Ratings**

| Symbol | Parameter | Minimum | Maximum | Units |
|---|---|---|---|---|
| $T_{STG}$ | Storage temperature | -40 | 125 | $^{o}C$ |
| $T_C$ | Case temperature under bias | -40 | 120 | $^{o}C$ |
| **Voltage with respect with ground** | | | | |
| $V_{DD}$ | 3.3V DC I/O supply voltage | -0.3 | 3.63 | V |
| $V_{IL}$ | Minimum signal input voltage | -0.5 | - | V |
| $V_{IH}$ | Maximum signal input voltage | - | -0.5 + 5.5 | V |

Table 17 contains the absolute maximum ratings for the Tsi340's ESD..

**Table 18: Absolute Maximum Ratings - ESD**

| Symbol | Parameter | Min | Max | Unit |
|---|---|---|---|---|
| $V_{ESD\_HBM}$ | Maximum ESD Voltage Discharge Tolerance for Human Body Model (HBM). [Test Conditions per JEDEC standard - JESD22-A114-B] | - | 2000 | V |
| $V_{ESD\_CDM}$ | Maximum ESD Voltage Discharge Tolerance for Charged Device Model (CDM). Test Conditions per JEDEC standard - JESD22-C101-A | - | 500 | V |

## 10.2 Recommended Operating Conditions

Table 19 contains the recommended operating conditions for the Tsi340.

**Table 19: Recommended Operating Conditions**

| Symbol | Parameter | Minimum | Maximum | Units | Notes |
|---|---|---|---|---|---|
| $V_{DD}$ | 3.3V DC I/O supply voltage | 3.0 | 3.6 | V | - |
| $T_A$ | Ambient temperature | 0.0 | 85 | $^oC$ | a, b |
| $T_{JUNC}$ | Junction temperature | 0.0 | 125 | $^oC$ | - |

a. No heat sink, no air flow.

b. Higher ambient temperatures are permissible provided $T_{JUNC}$ is not violated. For heat sink and air flow requirements for higher temperature operation, see .

## 10.3 Supply Current

Table 20 contains supply current characteristics for the Tsi340 for $V_{DD}$ at 3.6V. .

**Table 20: Supply Current Characteristics at 3.6V**

| | | | | |
|---|---|---|---|---|
| **Supply Current at 66MHz** | | | | |
| $I_{DD\ max}$ | Maximum $V_{DD}$ current draw | 715 | mA | a |
| $I_{DD\ idle}$ | Idle $V_{DD}$ current draw | 212 | mA | b |
| **Supply Current at 33MHz** | | | | |
| $I_{DD\ max}$ | Maximum $V_{DD}$ current draw | 343 | mA | a |
| $I_{DD\ idle}$ | Idle $V_{DD}$ current draw | 105 | mA | b |

a. Measured with 4 loads on secondary side with each bit on the bus switching.

b. No bus switching,

## 10.4 Power Supply Sequencing

The Tsi340 has only one voltage domain, so no special power sequencing is required.

## 10.5 DC Operating Characteristics

Table 21 contains DC operating characteristics for the Tsi340.

**Table 21: DC Operating Characteristics**

| Symbol | Parameter | Condition | Minimum | Maximum | Units | Notes |
|---|---|---|---|---|---|---|
| $V_{IL\_MISC}$ | Miscellaneous Input Low Voltage | - | -0.5 | $0.3V_{DD}$ | V | a |
| $V_{IH\_MISC}$ | Miscellaneous Input High Voltage | - | 2.0 | $V_{DD}$ + 0.5 | V | a |
| $V_{IL}$ | Input Low Voltage | - | -0.5 | $0.3V_{DD}$ | V | |
| $V_{IH}$ | PCI Input High Voltage | - | $0.5V_{DD}$ | -0.5 + 5.5 | V | |
| $V_{OL\_MISC}$ | Misc Output Low Voltage | $I_{OL}$ = 1.5mA | - | 0.5 | V | a |
| $V_{OH\_MISC}$ | Misc Output High Voltage | $I_{OH}$ = -0.5mA | $V_{DD}$ - 0.5 | - | V | a, b |
| $V_{OL}$ | Output Low Voltage | $I_{OL}$ = 1.5mA | - | $0.1V_{DD}$ | V | - |
| $V_{OH}$ | Output High Voltage | $I_{OH}$ = -0.5mA | $0.9V_{DD}$ | - | V | - |
| $C_{IN}$ | Input Pin Capacitance | - | - | 10 | pF | - |
| $C_{CLK}$ | Clock Pin Capacitance | - | 5 | 12 | pF | - |
| $C_{IDSEL}$ | IDSEL Pin Capacitance | - | - | 8 | pF | - |
| $L_{PIN}$ | Input Pin Inductance | - | - | 20 | nH | - |

a. Miscellaneous (Misc) signals include all 3.3V signals that are not PCI.

b. $V_{DD}$ = Min, I/O supply = Min.

# 10.6 AC Timing Specifications

This section discusses AC timing specifications for the Tsi340.

## 10.6.1 PCI Interface AC Timing

Table 22 contains the AC timing for the Tsi340.

**Table 22: AC Specifications for PCI Interface**

| Symbol | Parameter | PCI 66 | | PCI 33 | | Units | Notes |
|---|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | | |
| $T_{OV1}$ | Clock to Output Valid Delay for bused signals | 2 | 6 | 2 | 11 | ns | a, b, c |
| $T_{OV2}$ | Clock to Output Valid Delay for point to point signals | 2 | 6 | 2 | 12 | ns | a, b, c |
| $T_{OF}$ | Clock to Output Float Delay | - | 14 | - | 28 | ns | a, d |
| $T_{IS1}$ | Input Setup to clock for bused signals | 3 | - | 7 | - | ns | c, e, f |
| $T_{IS2}$ | Input Setup to clock for point to point signals | 5 | - | 10,12 | - | ns | c, d |
| $T_{IH1}$ | Input Hold time from clock | 0 | - | 0 | - | ns | d |
| $T_{RST}$ | Reset Active Time | 1 | - | 1 | - | ms | |
| $T_{RF}$ | Reset Active to output float delay | - | 40 | - | 40 | ns | - |

a. See the timing measurement conditions in Figure 8 on page 99.

b. See Figure 10 on page 99, Figure 11 on page 100, and Figure 12 on page 100.

c. Setup time for point-to-point signals applies to P[x]_REQ_B and P[x]_GNT_B only. All other signals are bused.

d. For purposes of Active/Float timing measurements, the HI-Z or “Off” state is defined to be when the total current delivered through the component pin is less than or equal to the leakage current specification.

e. See the timing measurement conditions in Figure 7 on page 98.

f. Setup time applies only when the device is not driving the pin. Devices cannot drive and receive signals at the same time.

### 10.6.2 PCI Clock (PCI_CLK) Specification

**Table 23: PCI Clock (PCI_CLK) Specification**

| Symbol | Parameter | PCI | | Units | Notes |
|---|---|---|---|---|---|
| | | Min | Max | | |
| $T_{C\_PCI33}$ | 33MHz PCI Clock Cycle Time | 30 | - | ns | - |
| $T_{CH\_PCI33}$ | 33MHz PCI Clock High Time | 11 | - | ns | - |
| $T_{CL\_PCI33}$ | 33MHz PCI Clock Low Time | 11 | - | ns | - |
| $T_{C\_PCI66}$ | 66MHz PCI Clock Cycle Time | 15 | 30 | ns | - |
| $T_{CH\_PCI66}$ | 66MHz PCI Clock High Time | 6 | - | ns | - |
| $T_{CL\_PCI66}$ | 66MHz PCI Clock Low Time | 6 | - | ns | - |
| $T_{SR\_PCI}$ | PCI Clock Slew Rate | 1 | 4 | V/ns | - |

## 10.7 AC Timing Waveforms

This section contains AC timing waveforms for the Tsi340.

**Figure 7: Input Timing Measurement Waveforms**

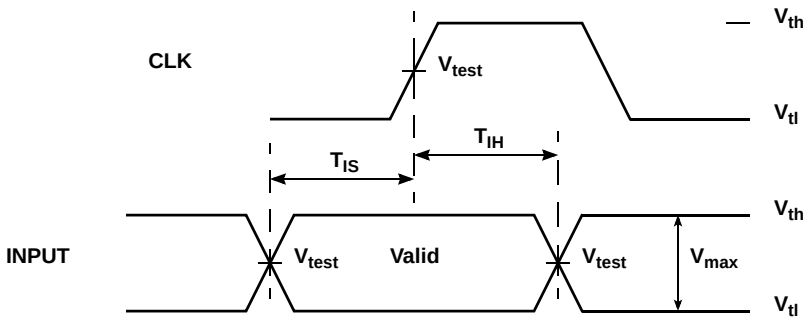


**Figure 8: Output Timing Measurement Waveforms**

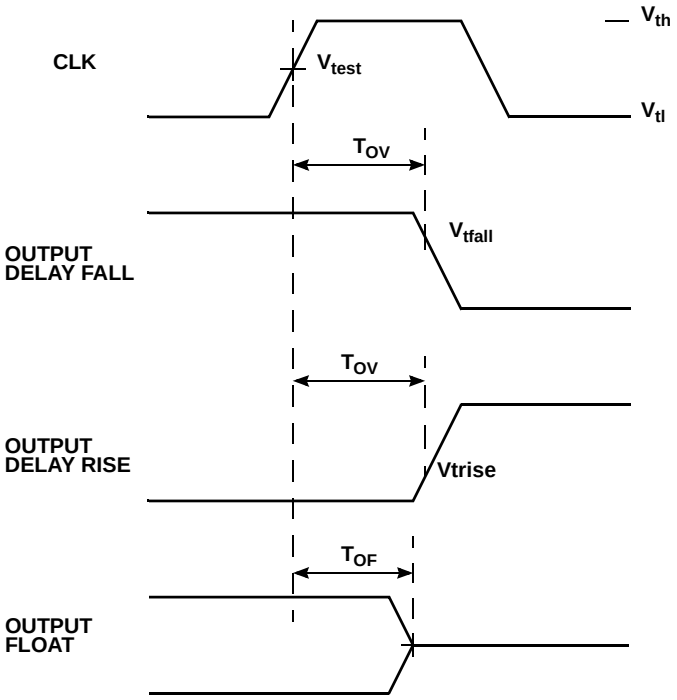



**Figure 9: AC Test Load for All Signals Except PCI**

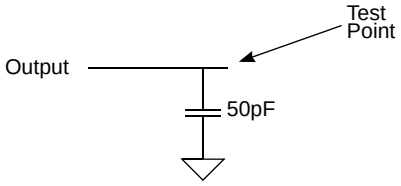



**Figure 10: PCI $T_{OV\ (max)}$ Rising Edge AC Test Load**

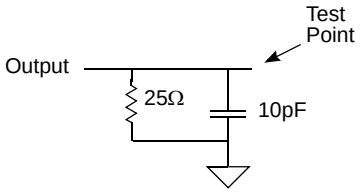


**Figure 11: PCI $T_{OV\ (max)}$ Falling Edge AC Test Load**

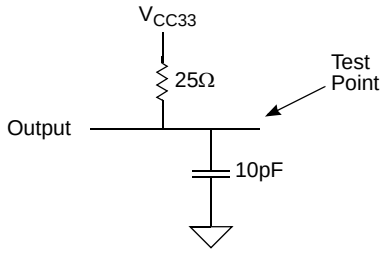
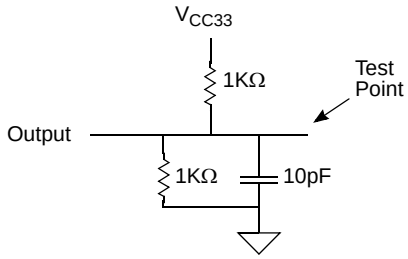



**Figure 12: PCI $T_{OV\ (min)}$ AC Test Load**



# 11. Register Descriptions

This chapter discusses the following topics about Tsi340's registers:



## 11.1 Overview

The following terms are used to describe Tsi340's register attributes:

- R – Read only. Can be initialized or modified by pin-straps or serial EEPROM.
- R/W – Read/write.
- R/W1C – Read/Write 1 to clear; writing a 0 has no effect. These register bits are only set by the Tsi340.
- RW1CS – Sticky Read Only, Write-1-to-Clear - Not initialized or modified by hot reset.
- R/WS Sticky Read / Write – Not initialized or modified by hot reset.
- R/W1S – Read 0/Write 1 to set (writing a 1 triggers an event such as an interrupt). These register bits are only cleared by the Tsi340.
- RC – Clear after read.
- RS – Sticky Read Only. Not initialized or modified by hot reset.
- Reserved – Do not write any value other than 0 to this field. Undefined data will be returned when read.
- Undefined – This value is undefined after reset because it is based on a bit setting, a pin setting, or a power-up setting.

## 11.2 PCI Configuration Space

This chapter provides a detailed description of Tsi340 configuration space registers.

Tsi340 configuration space uses the PCI-to-PCI bridge standard format specified in the *PCI-to- PCI Bridge Architecture Specification.* The header type at configuration address 0Eh reads as 01h, indicating that this device uses the PCI-to-PCI bridge format.

Tsi340 also contains device-specific registers, starting at address 40h. Use of these registers is not required for standard PCI-to-PCI bridge implementations.

### 11.2.1 Accessing Configuration Space Registers

The configuration space registers can be accessed only from the primary PCI bus. To access a register, perform a Type 0 format configuration read or write operation to that register. During the Type 0 address phase, P_AD<7:2> indicates the Dword offset of the register. During the data phase,

Software changes the configuration register values that affect Tsi340 behavior only during initialization. Change these values subsequently only when both the primary and secondary PCI buses are idle, and the data buffers are empty; otherwise, the behavior of Tsi340 is unpredictable.

P_CBE_b<3:0> selects the bytes in the Dword that is being accessed.

## 11.3 Register Map

The following table lists the register map for the Tsi340.

**Table 24: Register Map**

| Offset | Name | Bits | See |
|---|---|---|---|
| 0x00 | Device ID | 31:16 | “Vendor ID Register—Offset 0x00” on page 107 |
| 0x00 | Vendor ID | 15:0 | “Device ID Register—Offset 0x00” on page 107 |
| 0x04 | Primary Status Register | 31:16 | “Primary Status Register—Offset 0x04” on page 110 |
| 0x04 | Primary Command Register | 15:0 | “Primary Command Register—Offset 0x04” on page 108 |
| 0x08 | Base Class Code | 31:24 | “Base Class Code Register—Offset 0x08” on page 111 |
| 0x08 | Subclass Code | 23:16 | “Subclass Code Register—Offset 0x08” on page 111 |
| 0x08 | Programming Interface Register | 15:8 | “Programming Interface Register—Offset 08” on page 111 |
| 0x08 | Revision ID | 7:0 | “Revision ID Register—Offset 0x08” on page 111 |
| 0x0C | Reserved | - | - |
| 0x0C | Header Type | 23:16 | “Header Type Register—Offset 0x0C” on page 112 |
| 0x0C | Primary Latency Timer | 15:8 | “Primary Latency Timer Register—Offset 0x0C” on page 112 |
| 0x0C | Cache Line Size | 7:0 | “Cache Line Size Register—Offset 0x0C” on page 112 |
| 0x10-0x14 | Reserved | - | - |
| 0x18 | Secondary latency timer | 31:24 | “Secondary Latency Timer Register—Offset 0x18” on page 114 |
| 0x18 | Subordinate Bus Number | 23:16 | “Subordinate Bus Number Register—Offset 0x18” on page 113 |
| 0x18 | Secondary Bus Number | 15:8 | “Secondary Bus Number Register—Offset 0x18” on page 113 |
| 0x18 | Primary Bus Number | 7:0 | “Primary Bus Number Register—Offset 0x18” on page 113 |
| 0x1C | Secondary Status | 31:16 | “Secondary Status Register—Offset 0x1C” on page 116 |
| 0x1C | I/O Limit | 15:8 | “I/O Limit Address Register—Offset 0x1C” on page 115 |
| 0x1C | I/O Base | 7:0 | “I/O Base Address Register—Offset 0x1C” on page 115 |
| 0x20 | Memory Limit Address | 31:16 | “Memory Limit Address Register—Offset 0x20” on page 117 |
| 0x20 | Memory Base Address | 15:0 | “Memory Base Address Register—Offset 0x20” on page 117 |

**Table 24: Register Map *(Continued)***

| Offset | Name | Bits | See |
|---|---|---|---|
| 0x24 | Prefetchable Memory Limit | 31:16 | “Prefetchable Memory Limit Address Register—Offset 0x24” on page 118 |
| 0x24 | Prefetchable Memory Base | 15:0 | “ Prefetchable Memory Base Address Register—Offset 0x24” on page 118 |
| 0x28 | Prefetchable Memory Base Upper 32 Bits | 31:0 | “ Prefetchable Memory Base Address Upper 32 Bits Register—Offset 0x28” on page 119 |
| 0x2C | Prefetchable Memory Limit Upper 32 Bits | 31:0 | “Prefetchable Memory Limit Address Upper 32 Bits Register—Offset 0x2C” on page 119 |
| 0x30 | I/O Limit Upper 16 Bits | 31:16 | “I/O Limit Address Upper 16 Bits Register—Offset 0x30” on page 120 |
| 0x30 | I/O Base Upper 16 Bits | 15:0 | “I/O Base Address Upper 16 Bits Register—Offset 0x30” on page 120 |
| 0x34 | Reserved | - | - |
| 0x34 | ECP Pointer | 7:0 | “ECP Pointer Register—Offset 0x34” on page 121 |
| 0x38 | Reserved | - | - |
| 0x3C | Bridge Control | 31:16 | “Bridge Control Register—Offset 0x3C” on page 122 |
| 0x3C | Interrupt Pin | 15:8 | “Interrupt Pin Register—Offset 0x3C” on page 121 |
| 0x3C | Interrupt Line | 7:0 | “Interrupt Line Register – Offset 0x3C” on page 121 |
| 0x40 | Subsystem ID | 31:16 | “Subsystem ID Register — Offset 0x40 ” on page 125 |
| 0x40 | Subsystem Vendor ID | 15:0 | “Subsystem Vendor ID Register — Offset 0x40 ” on page 125 |
| 0x44 | Arbiter Control | 31:16 | “Arbiter Control Register—Offset 0x44” on page 127 |
| 0x44 | Diagnostic Control | 15:8 | “Chip Control Register/Diagnostic Control — Offset 0x44” on page 126 |
| 0x44 | Chip Control | 7:0 | “Chip Control Register/Diagnostic Control — Offset 0x44” on page 126 |
| 0x48 | Reserved | - | - |
| 0x48 | Memory Read Control | 7:0 | “Memory Read Control Register — Offset 0x48” on page 128 |
| 0x4C | Secondary Bus Arbiter Preemption Control | 31:24 | “Secondary Bus Arbiter Preemption Control Register — Offset 0x4C” on page 129 |
| 0x4C | Reserved | - | - |
| 0x50-0x60 | Reserved | - | - |

**Table 24: Register Map *(Continued)***

| Offset | Name | Bits | See |
|---|---|---|---|
| 0x64 | Reserved | - | - |
| 0x64 | P_SERR_b Event Disable | 7:0 | “P_SERR_b Event Disable Register—Offset 0x64” on page 130 |
| 0x68 | Reserved | - | - |
| 0x68 | P_SERR_b Status | 23:16 | “P_SERR_b Status Register — Offset 0x68” on page 132 |
| 0x68 | Secondary Clock Control | 15:0 | “Secondary Clock Control Register—Offset 0x68” on page 131 |
| 0x6C | CLKRUN | 31:24 | “CLKRUN Register — Offset 0x6C” on page 133 |
| 0x6C | Reserved | - | - |
| 0x70 | Reserved | - | - |
| 0x74 | Reserved | - | - |
| 0x74 | Port Option | 15:0 | “Port Option Register — Offset 0x74” on page 134 |
| 0x78-0x7C | Reserved | - | - |
| 0x80 | Power Management Capabilities | 31:16 | “Power Management Data Register—Offset 0x84” on page 137 |
| 0x80 | Next Item Pointer | 15:8 | “Next Item Pointer Register—Offset 0x80” on page 136 |
| 0x80 | Capability ID | 7:0 | “Capability ID Register—Offset 0x80” on page 135 |
| 0x84 | Reserved | - | - |
| 0x84 | PMCSR_BSE | 23:16 | “PMCSR_BSE Register — Offset 0x84” on page 138 |
| 0x84 | Power Management CSR | 15:0 | “Power Management Data Register—Offset 0x84” on page 137 |
| 0x88-0x8C | Reserved | - | - |
| 0x90 | Reserved | - | - |
| 0x90 | Hot Swap Control Status | 23;16 | “HS Control Status Register — Offset 0x90” on page 138 |
| 0x90 | HS Next Item Pointer | 15:8 | “HS Next Item Pointer Register — Offset 0x90” on page 138 |
| 0x90 | HS Capability ID | 7:0 | “HS Capability ID Register — Offset 0x90” on page 138 |
| 0x94-0xBC | Reserved | - | - |
| 0xC0 | Reserved | - | - |

**Table 24: Register Map *(Continued)***

| Offset | Name | Bits | See |
|---|---|---|---|
| 0xC0 | Miscellaneous Control | | “Miscellaneous Control Register — Offset 0xC0” on page 140 |
| 0xC0 | Reserved | - | - |
| 0xC4-0xF0 | Reserved | - | - |
| 0xF4-0xFF | Reserved | - | - |

# 11.4 PCI-to-PCI Bridge Standard Register Descriptions

Tsi340 configuration space uses the PCI-to-PCI bridge standard format specified in the *PCI-to- PCI Bridge Architecture Specification.* The header type at configuration address 0x0E reads as 0x01, indicating that this device uses the PCI-to-PCI bridge format.

## 11.4.1 Vendor ID Register—Offset 0x00

This section describes the vendor ID register.

Byte enable P_CBE_b<3:0> = xx00b

| Bits | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 15:0 | Vendor ID | Identifies the vendor of this device.<br>Internally hardwired to be 0x10E3 | R | 0x10E3 |

## 11.4.2 Device ID Register—Offset 0x00

This section describes the device ID register.

Byte enable P_CBE_b<3:0> = 00xxb

| Bits | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 31:16 | Device ID | Identifies this device as Tsi340. | R | 0x8140 |

### 11.4.3 Primary Command Register—Offset 0x04

This section describes the primary command register.

These bits affect the behavior of Tsi340 primary interface, except where noted. Some of the bits are repeated in the bridge control register, to act on the secondary interface.

This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = xx00b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 0 | I/O space enable | Controls Tsi340's response to I/O transactions on the primary interface.<br>0 = Tsi340 does not respond to I/O transactions initiated on the primary bus.<br>1 = Tsi340 response to I/O transactions initiated on the primary bus is enabled. | R/W | 0x0 |
| 1 | Memory space enable | Controls Tsi340's response to memory transactions on Tsi340 primary interface.<br>0 = Tsi340 does not respond to memory transactions initiated on the primary bus.<br>1 = Tsi340 response to memory transactions initiated on the primary bus is enabled. | R/W | 0x0 |
| 2 | Master enable | Controls Tsi340's ability to initiate memory and I/O transactions on the primary bus on behalf of an initiator on the secondary bus. Forwarding of configuration transactions is not affected.<br>0 = Tsi340 does not respond to I/O or memory transactions on the secondary interface and does not initiate I/O or memory transactions on the primary interface.<br>1 = Tsi340 is enabled to operate as an initiator on the primary bus and responds to I/O and memory transactions initiated on the secondary bus. | R/W | 0x0 |
| 3 | Special cycle enable | Tsi340 ignores special cycle transactions, so this bit is read only and returns 0. | R | 0x0 |
| 4 | Memory write and invalidate enable | Tsi340 generates memory write and invalidate transactions only when operating on behalf of another master whose memory write and invalidate transaction is crossing Tsi340. | R | 0x0 |

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 5 | VGA snoop enable | Controls Tsi340's response to VGA compatible palette write transactions. VGA palette write transactions correspond to I/O transactions whose address bits are as follows:<br>• P_AD<9:0> are equal to 3C6h, 3C8h, and 3C9h.<br>• P_AD<15:10> are not decoded.<br>• P_AD<31:16> must be 0.<br>0 = VGA palette write transactions on the primary interface are ignored unless they fall inside Tsi340's I/O address range.<br>1 = VGA palette write transactions on the primary interface are positively decoded and forwarded to the secondary interface. | R/W | 0x0 |
| 6 | Parity error response | Controls Tsi340's response when a parity error is detected on the primary interface.<br>0 = Tsi340 does not assert P_PERR_b, nor does it set the data parity reported bit in the status register. Tsi340 does not report address parity errors by asserting P_SERR_b.<br>1 = Tsi340 drives P_PERR_b and<br>conditionally sets the data parity reported bit in the status register when a data parity error is detected. Tsi340 allows P_SERR_b assertion when address parity errors are detected on the primary interface. | R/W | 0x0 |
| 7 | Wait cycle control | Reads as 0 to indicate that Tsi340 does not perform address or data stepping. | R | 0x0 |
| 8 | SERR# enable | Controls the enable for P_SERR_b on the primary interface.<br>0 = Signal P_SERR_b cannot be driven by Tsi340.<br>1 = Signal P_SERR_b can be driven low by Tsi340. | R/W | 0x0 |
| 9 | Fast back-to-back enable | Reads as 0 to indicate that Tsi340 does not generate fast back-to-back transactions on the primary bus. | R | 0x0 |
| 15:10 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |

### 11.4.4 Primary Status Register—Offset 0x04

This section describes the primary status register.

These bits reflect the status of Tsi340 primary interface. Bits reflecting the status of the secondary interface are found in the secondary status register. W1TC indicates that writing 1 to a bit sets that bit to 0. Writing 0 has no effect.

Byte enable P_CBE_b<3:0> = 00xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 19:16 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 20 | ECP | Enhanced Capabilities Port (ECP) enable. Reads as 1 in Tsi340 to indicate that Tsi340 supports an enhanced capabilities list. | R | 0x1 |
| 21 | 66-MHz capable | Set to 1 to indicates that primary interface is 66- MHz capable. | R | 0x1 |
| 22 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 23 | Fast back-to-back capable | Indicates that Tsi340 is able to respond to fast back-to-back transactions on the primary interface. | R | 0x1 |
| 24 | Data parity detected | This bit is set to 1 when all of the following are true:<br>• Tsi340 is a master on the primary bus.<br>• Signal P_PERR_b is detected asserted, or a parity error is detected on the primary bus.<br>• The parity error response bit is set in the command register. | R/W1TC | 0x0 |
| 26:25 | DEVSEL# timing | Indicates slowest response to a Non configuration command on the primary interface. Reads as 01b to indicate that Tsi340 responds no slower than with medium timing. | R | 01b |
| 27 | Signaled target abort | This bit is set to 1 when Tsi340 is acting as a target on the primary bus and returns a target abort to the primary master. | R/W1TC | 0x0 |
| 28 | Received target abort | This bit is set to 1 when Tsi340 is acting as a master on the primary bus and receives a target abort from the primary target. | R/W1TC | 0x0 |
| 29 | Received master abort | This bit is set to 1 when Tsi340 is acting as a master on the primary bus and receives a master abort. | R/W1TC | 0x0 |
| 30 | Signaled system error | This bit is set to 1 when Tsi340 has asserted P_SERR_b. | R/W1TC | 0x0 |
| 31 | Detected parity error | This bit is set to 1 when Tsi340 detects an address or data parity error on the primary interface. | R/W1TC | 0x0 |

### 11.4.5 Revision ID Register—Offset 0x08

This section describes the revision ID register.

Byte enable P_CBE_b<3:0> = xxx0b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 7:0 | Revision ID | Indicates the revision number of this device.<br>The initial revision reads as 0x01 | R | 0x01 |

### 11.4.6 Programming Interface Register—Offset 08

This section describes the programming interface register.

Byte enable P_CBE_b<3:0> = xx0xb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 15:8 | Programming interface | No programming interfaces have been defined for PCI-to-PCI bridges. | R | 0x0 |

### 11.4.7 Subclass Code Register—Offset 0x08

This section describes the subclass code register.

Byte enable P_CBE_b<:0> = x0xxb

| Bit | Name | Description | R/W | Reset Value |
|---|---|---|---|---|
| 23:16 | Subclass code | Indicates that this bridge device is a PCI-to-PCI bridge. | R | 0x04 |

### 11.4.8 Base Class Code Register—Offset 0x08

This section describes the base class code register.

Byte enable P_CBE_b<3> = 0xxxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 31:24 | Base class code | Indicate this device is a bridge device. | R | 0x06 |

### 11.4.9 Cache Line Size Register—Offset 0x0C

This section describes the cache line size register.

Byte enable P_CBE_b<3:0> = xxx0b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 7:0 | Cache line size | Designates the cache line size for the system in units of 32-bit Dwords. Used for prefetching memory read transactions and for terminating memory write and invalidate transactions.<br>The cache line size should be written as a power of 2. If the value is not a power of 2 or is greater than 16, Tsi340 behaves as if the cache line size were 16. | R/W | 0x0 |

### 11.4.10 Primary Latency Timer Register—Offset 0x0C

This section describes the primary latency timer register.

Byte enable P_CBE_b<3:0> = xx0xb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 15:11 | Master latency timer | Master latency timer for the primary interface.<br>Indicates the number of PCI clock cycles from the assertion of P_FRAME_b to the expiration of the timer when Tsi340 is acting as a master on the primary interface. All bits are writable, resulting in a granularity of eight PCI clock cycle.<br>0 = Tsi340 relinquishes the bus after the first data transfer when Tsi340's primary bus grant has been deasserted, with the exception of memory write and invalidate transactions. | R/W | 0x0 |
| 10:8 | Reserved | Set to b'000' to force 8-cycle increments for the latency timer. | R | 0x0 |

### 11.4.11 Header Type Register—Offset 0x0C

This section describes the header type register.

Byte enable P_CBE_b<3:0> = x0xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 23:16 | Header type | Defines the layout of addresses 0x10 through 0x3F in configuration space.<br>Reads as 0x01 to indicate that the register layout conforms to the standard PCI-to-PCI bridge layout. | R | 0x01 |

### 11.4.12 Primary Bus Number Register—Offset 0x18

This section describes the primary bus number register.This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = xxx0b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 7:0 | Primary bus number | Indicates the number of the PCI bus to which the primary interface is connected. Tsi340 uses this register to decode Type 1 configuration transactions on the secondary interface that should either be converted to special cycle transactions on the primary interface or passed upstream unaltered. | R/W | 0x0 |

### 11.4.13 Secondary Bus Number Register—Offset 0x18

This section describes the secondary bus number register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0>= xx0xb

| Bit | Name | Description | R/W | Reset Value |
|---|---|---|---|---|
| 15:8 | Secondary bus number | Indicates the number of the PCI bus to which the secondary interface is connected. Tsi340 uses this register to determine when to respond to and forward Type 1 configuration transactions on the primary interface, and to determine when to convert them to Type 0 or special cycle transactions on the secondary interface. | R/W | 0x0 |

### 11.4.14 Subordinate Bus Number Register—Offset 0x18

This section describes the subordinate bus number register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0>= x0xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 23:16 | Subordinate bus number | Indicates the number of the highest numbered PCI bus that is behind (or subordinate to) Tsi340. Used in conjunction with the secondary bus number to determine when to respond to Type 1 configuration transactions on the primary interface and pass them to the secondary interface as a Type 1 configuration transaction. | R/W | 0x0 |

### 11.4.15 Secondary Latency Timer Register—Offset 0x18

This section describes the secondary latency timer register.

Byte enable P_CBE_b<3:0> = 0xxxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 31:27 | Secondary latency timer | Master latency timer for the secondary interface. Indicates the number of PCI clock cycles from the assertion of S_FRAME_b to the expiration of the timer when Tsi340 is acting as a master on the secondary interface.<br>All bits are writable, resulting in a granularity of eight PCI clock cycle.<br>When 0—Tsi340 ends the transaction after the first data transfer when Tsi340's secondary bus grant has been deasserted, with the exception of memory write and invalidate transactions. | R/W | 0x0 |
| 26:24 | Reserved | Set to b'000' to force 8-cycle increments for the latency timer. | R | 0x0 |

### 11.4.16 I/O Base Address Register—Offset 0x1C

This section describes the I/O base address register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = xxx0b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 3:0 | 32-bit indicator | The low 4 bits of this register read as 0x01 to indicate that Tsi340 supports 32-bit I/O address decoding. | R | 0x01 |
| 7:4 | I/O base address [15:12] | Defines the bottom address of an address range used by Tsi340 to determine when to forward I/O transactions from one interface to the other. The upper 4 bits are write-able and correspond to address bits <15:12>. The lower 12 bits of the address are assumed 0. The upper 16 bits corresponding to address bits <31:16> are defined in the I/O base address upper 16 bits register. The I/O address range adheres to 4 KB alignment and granularity. | R/W | 0x0 |

### 11.4.17 I/O Limit Address Register—Offset 0x1C

This section describes the I/O limit address register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = xx0xb

| Bit | Name | Description | R/W | Reset Value |
|---|---|---|---|---|
| 11:8 | 32-bit indicator | The low 4 bits of this register read as 0x01 to indicate that Tsi340 supports 32-bit I/O address decoding. | R | 0x01 |
| 15:12 | I/O limit address[15:12] | Defines the top address of an address range used by Tsi340 to determine when to forward I/O transactions from one interface to the other. The upper 4 bits are writable and correspond to address bits <15:12>. The lower 12 bits of the address are assumed 0xFFF. The upper 16 bits register corresponding to address bits <31:16> are defined in the I/O limit address upper 16 bits register. The I/O address range adheres to 4 KB alignment and granularity. | R/W | 0x0 |

### 11.4.18 Secondary Status Register—Offset 0x1C

This section describes the secondary status register.

These bits reflect the status of Tsi340 secondary interface. W1TC indicates that writing 1 to that bit sets the bit to 0. Writing 0 has no effect.

Byte enable P_CBE_b<3:0> = 00xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 20:16 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 21 | 66-MHz capable | Indicates that secondary interface is 66- MHz capable. | R | 0x1 |
| 22 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 23 | Fast back-to-back capable | Indicates that Tsi340 is able to respond to fast back-to-back transactions on the secondary interface. | R | 0x1 |
| 24 | Data parity detected | This bit is set to 1 when all of the following are true:<br>• Tsi340 is a master on the secondary bus.<br>• Signal S_PERR_b is detected asserted, or a parity error is detected on the primary bus.<br>• The parity error response bit is set in the bridge control register. | R/W1TC | 0x0 |
| 26:25 | DEVSEL# timing | Indicates slowest response to a Non configuration command on the secondary interface.<br>Reads as 01b to indicate that Tsi340 responds no slower than with medium timing. | R | 01b |
| 27 | Signaled target abort | This bit is set to 1 when Tsi340 is acting as a target on the secondary bus and returns a target abort to the secondary bus master. | R/W1TC | 0x0 |
| 28 | Received target abort | This bit is set to 1 when Tsi340 is acting as a master on the secondary bus and receives a target abort from the secondary bus target. | R/W1TC | 0x0 |
| 29 | Received master abort | This bit is set to 1 when Tsi340 is acting as an initiator on the secondary bus and receives a master abort. | R/W1TC | 0x0 |
| 30 | Received system error | This bit is set to 1 when Tsi340 detects the assertion of S_SERR_b on the secondary interface. | R/W1TC | 0x0 |
| 31 | Detected parity error | This bit is set to 1 when Tsi340 detects an address or data parity error on the secondary interface. | R/W1TC | 0x0 |

### 11.4.19 Memory Base Address Register—Offset 0x20

This section describes the memory base address register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = xx00b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 3:0 | Reserved | The lower 4 bits of this register are read only and return 0. | R | 0x0 |
| 15:4 | Memory base address <31:20> | Defines the bottom address of an address range used by Tsi340 to determine when to forward memory transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed to be 0. The memory address range adheres to 1MB alignment and granularity. | R/W | 0x0 |

### 11.4.20 Memory Limit Address Register—Offset 0x20

This section describes the memory limit address register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = 00xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 19:16 | Reserved | The lower 4 bits of this register are read only and return 0. | R | 0x0 |
| 31:20 | Memory limit address <31:20> | Defines the top address of an address range used by Tsi340 to determine when to forward memory transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed FFFFFh. The memory address range adheres to 1MB alignment and granularity. | R/W | 0x0 |

### 11.4.21 Prefetchable Memory Base Address Register—Offset 0x24

This section describes the prefetchable memory base address register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = xx00b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 3:0 | 64-bit indicator | The low 4 bits of this register are read only and return 0x1 to indicate that this range supports 64-bit addressing. | R | 0x1 |
| 15:4 | Prefetchable memory base address <31:20> | Defines the bottom address of an address range used by Tsi340 to determine when to forward memory read and write transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>. The lower 20 bits of the address are assumed 0. The memory base register upper 32 bits contain the upper half of the base address. The memory address range adheres to 1MB alignment and granularity. | R/W | 0x0 |

### 11.4.22 Prefetchable Memory Limit Address Register—Offset 0x24

This section describes the prefetchable memory limit address register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = 00xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 19:16 | 64-bit indicator | The low 4 bits of this register are read only and return 0x1 to indicate that this range supports 64-bit addressing. | R | 0x1 |
| 31:20 | Prefetchable memory limit address <31:20> | Defines the top address of an address range used by Tsi340 to determine when to forward memory read and write transactions from one interface to the other. The upper 12 bits are writable and correspond to address bits <31:20>.<br>The lower 20 bits of the address are assumed to be 0xFFFFF. The memory limit upper 32 bits register contains the upper half of the limit address. The memory address range adheres to 1MB alignment and granularity. | R/W | 0x0 |

### 11.4.23 Prefetchable Memory Base Address Upper 32 Bits Register—Offset 0x28

This section describes the prefetchable memory base address upper 32 bits register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0>= 0000b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 31:0 | Upper 32 prefetchable memory base address <63:32> | Defines the upper 32 bits of a 64-bit bottom address of an address range used by Tsi340 to determine when to forward memory read and write transactions from one interface to the other. The memory address range adheres to 1MB alignment and granularity. | R/W | 0x0 |

### 11.4.24 Prefetchable Memory Limit Address Upper 32 Bits Register—Offset 0x2C

This section describes the prefetchable memory limit address upper 32 bits register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = 0000b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 31:0 | Upper 32 prefetchable memory limit address <63:32> | Defines the upper 32 bits of a 64-bit top address of an address range used by the Tsi340 to determine when to forward memory read and write transactions from one interface to the other. Extra read transactions should have no side effects. The memory address range adheres to 1MB alignment and granularity. | R/W | 0x0 |

### 11.4.25 I/O Base Address Upper 16 Bits Register—Offset 0x30

This section describes the I/O base address upper 16 bits register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = xx00b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 15:0 | I/O base address upper 16 bits <31:16> | Defines the upper 16 bits of a 32-bit bottom address of an address range used by Tsi340 to determine when to forward I/O transactions from one interface to the other. The I/O address range adheres to 4KB alignment and granularity. | R/W | 0x0 |

### 11.4.26 I/O Limit Address Upper 16 Bits Register—Offset 0x30

This section describes the I/O limit address upper 16 bits register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0> = 00xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 31:16 | I/O limit address upper 16 bits <31:16> | Defines the upper 16 bits of a 32-bit top address of an address range used by Tsi340 to determine when to forward I/O transactions from one interface to the other. The I/O address range adheres to 4KB alignment and granularity. | R/W | 0x0 |

### 11.4.27 ECP Pointer Register—Offset 0x34

This section describes the ECP pointer register.

Byte enable P_CBE_b<3:0> = 0000b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 7:0 | ECP_PTR | Enhanced Capabilities Port (ECP) offset pointer. Reads as 0x80 in Tsi340 to indicate that the first item, which corresponds to the power management registers, resides at that configuration offset. | R | 0x80 |
| 31:8 | Reserved | Reserved. Return 0 when read. | R | 0x0 |

### 11.4.28 Interrupt Line Register – Offset 0x3C

This section describes the interrupt pin register.

Byte enable P_CBE_b<3:0> = xxx0b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 7:0 | Interrupt Line | Tsi340 does not implement an interrupt signal. | R/W | 0xFF |

### 11.4.29 Interrupt Pin Register—Offset 0x3C

This section describes the interrupt pin register.

Byte enable P_CBE_b<3:0> = xx0xb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 15:8 | Interrupt pin | Reads as 0 to indicate that Tsi340 does not have an interrupt pin. | R | 0x0 |

### 11.4.30 Bridge Control Register—Offset 0x3C

This section describes the bridge control register. This register must be initialized by configuration software.

Byte enable P_CBE_b<3:0>= 00xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 16 | Parity error response | Controls Tsi340's response when a parity error is detected on the secondary interface.<br>0 = Tsi340 does not assert S_PERR_b, nor does it set the data parity reported bit in the secondary status register. Tsi340 does not report address parity errors by asserting P_SERR_b.<br>1 = Tsi340 drives S_PERR_b and conditionally sets the data parity reported bit in the secondary status register when a data parity error is detected on the secondary interface (see Section 7.0). Also must be set to 1 to allow P_SERR_b assertion when address parity errors are detected on the secondary interface. | R/W | 0x0 |
| 17 | SERR# forward enable | Controls whether Tsi340 asserts P_SERR_b when it detects S_SERR_b asserted.<br>0 = Tsi340 does not drive P_SERR_b in response to S_SERR_b assertion.<br>1 = Tsi340 asserts P_SERR_b when S_SERR_b is detected asserted (the primary SERR# driver enable bit must also be set). | R/W | 0x0 |
| 18 | ISA enable | Modifies Tsi340's response to ISA I/O addresses. Applies only to those addresses falling within the I/O base and limit address registers and within the first 64KB of PCI I/O space.<br>0 = Tsi340 forwards all I/O transactions downstream that fall within the I/O base and limit address registers.<br>1 = Tsi340 ignores primary bus I/O transactions within the I/O base and limit address registers and within the first 64KB of PCI I/O space that address the last 768 bytes in each 1KB block. Secondary bus I/O transactions are forwarded upstream if the address falls within the last 768 bytes in each 1KB block. | R/W | 0x0 |

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 19 | VGA enable | Modifies Tsi340's response to VGA compatible addresses.<br>0 = VGA transactions are ignored on the primary bus unless they fall within the I/O base and limit address registers and the ISA mode is 0.<br>1 = Tsi340 positively decodes and forwards the following transactions downstream, regardless of the values of the I/O base and limit registers, ISA mode bit, or VGA snoop bit:<br>• Memory transactions addressing<br>0x000A0000–000BFFFF<br>• I/O transactions addressing:<br>— P_AD<9:0> = 3B0h–3BBh and 3C0h–3DFh<br>— P_AD<15:10> are not decoded.<br>— P_AD<31:16> = 0x0000<br>I/O and memory space enable bits must be set in the command register.<br>The transactions listed here are ignored by Tsi340 on the secondary bus. | R/W | 0x0 |
| 20 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 21 | Master abort mode | Controls Tsi340's behavior when a master abort termination occurs in response to a transaction initiated by Tsi340 on either the primary or secondary PCI interface.<br>0 = Tsi340 asserts TRDY# on the initiator bus for delayed transactions, and 0xFFFF FFFF for read transactions. For posted write transactions, P_SERR_b is not asserted.<br>1 = Tsi340 returns a target abort on the initiator bus for delayed transactions. For posted write transactions, Tsi340 asserts P_SERR_b if the SERR# enable bit is set in the command register. | R/W | 0x0 |
| 22 | Secondary bus reset | Controls S_RST_b on the secondary interface.<br>0 = Tsi340 deasserts S_RST_b.<br>1 = Tsi340 asserts S_RST_b. When S_RST_b is asserted, the data buffers and the secondary interface are initialized back to reset conditions. The primary interface and configuration registers are not affected by the assertion of S_RST_b. | R/W | 0x0 |
| 23 | Fast back-to-back enable | Reads as 0 to indicate that Tsi340 does not generate fast back-to-back transactions on the secondary bus | R | 0x0 |

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 24 | Primary master timeout | Sets the maximum number of PCI clock cycles that Tsi340 waits for an initiator on the primary bus to repeat a delayed transaction request. The counter starts once the delayed transaction completion is at the head of the queue. If the master has not repeated the transaction at least once before the counter expires, Tsi340 discards the transaction from its queues.<br>0 = The primary master timeout value is 2^15 PCI clock cycles.<br>1 = The value is 2^10 PCI clock. | R/W | 0x0 |
| 25 | Secondary master timeout | Sets the maximum number of PCI clock cycles that Tsi340 waits for an initiator on the secondary bus to repeat a delayed transaction request. The counter starts once the delayed transaction completion is at the head of the queue. If the master has not repeated the transaction at least once before the counter expires, Tsi340 discards the transaction from its queues.<br>0 = The primary master timeout value is 2^15 PCI clock cycles.<br>1 = The value is 2^10 PCI clock cycles. | R/W | 0x0 |
| 26 | Master timeout status | This bit is set to 1 when either the primary master timeout counter or the secondary master timeout counter expires and a delayed transaction is discarded from Tsi340's queues. Write 1 to clear. | R/W1TC | 0x0 |
| 27 | Master timeout SERR# enable | Controls assertion of P_SERR_b during a master timeout.<br>0 = Signal P_SERR_b is not asserted as a result of a master timeout.<br>1 = Signal P_SERR_b is asserted when either the primary master timeout counter or the secondary master timeout counter expires and a delayed transaction is discarded from Tsi340's queues. The SERR# enable bit in the command register must also be set. | R/W | 0x0 |
| 31:28 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |

# 11.5 Device Specific Register Descriptions

Tsi340 contains device-specific registers, starting at address 40h. Use of these registers is not required for standard PCI-to-PCI bridge implementations.

## 11.5.1 Subsystem Vendor ID Register — Offset 0x40

Byte enable P_CBE_b<3:0> = xx00b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 15:0 | Subsystem Vendor ID | Subsystem Vendor ID for add-in card manufacturers. | R/W | 0x0 |

## 11.5.2 Subsystem ID Register — Offset 0x40

Byte enable P_CBE_b<3:0> = 00xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 31:16 | Subsystem ID | Subsystem ID for add-in card manufacturers. | R/W | 0x0 |

### 11.5.3 Chip Control Register/Diagnostic Control — Offset 0x44

This section describes the chip control register.

Byte enable P_CBE_b<3:0> = xx00b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 0 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 1 | Memory write disconnect control | Controls when Tsi340, as a target, disconnects memory write transactions.<br>0 = Tsi340 disconnects on queue full or on a 4KB boundary.<br>1 = Tsi340 disconnects on a cache line boundary, as well as when the queue fills or on a 4KB boundary. | R/W | 0x0 |
| 3:2 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 4 | Secondary bus prefetch disable | Controls Tsi340's ability to prefetch during upstream memory read transactions.<br>0 = Tsi340 prefetches and does not forward byte enable bits during<br>1 = Tsi340 requests only one Dword from the target during memory read transactions and forwards read byte enable bits. Tsi340 returns a target disconnect to the requesting master on the first data transfer. Memory read line and memory read multiple transactions are still prefetchable. | R/W | 0x0 |
| 7:5 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 8 | Chip reset | Chip and secondary bus reset control.<br>1 = Causes Tsi340 to perform a chip reset. Data buffers, configuration registers, and both the primary and secondary interfaces are reset to their initial state. Tsi340 clears this bit once chip reset is complete. Tsi340 can then be reconfigured.<br>Secondary bus reset S_RST_b is asserted and the secondary reset bit in the bridge control register is set when this bit is set. The secondary reset bit in the bridge control Register must be cleared in order to deassert S_RST_b. | R/W1TR | 0x0 |
| 15:9 | Reserved | Reserved. Returns 0 when read | R | 0x0 |

### 11.5.4 Arbiter Control Register—Offset 0x44

This section describes the arbiter control register.

Byte enableP_CBE_b<3:0> = 00xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 19:16 | Arbiter control | Each bit controls whether a secondary bus master is assigned to the high priority arbiter group or the low priority arbiter group. Bits <19:16> correspond to request inputs S_REQ_b<3:0>, respectively.<br>0 = Indicates that the master belongs to the low priority group.<br>1 = Indicates that the master belongs to the high priority group. | R/W | 0x0 |
| 24:20 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 25 | Priority of Secondary Interface | Controls whether the secondary interface of the bridge is in the high priority group or the low priority group.<br>0 = low priority<br>1 = high priority | RW | 0x1 |
| 31:26 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |

### 11.5.5 Memory Read Control Register — Offset 0x48

Byte enableP_CBE_b<3:0> = xxx0b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 7:6 | Downstream Transaction Maximum Prefetch count | This register bits designates the maximum prefetch count for downstream Memory Read transactions.<br>00 = 16 Dwords<br>01 = 32 Dwords<br>10 = 48 Dwords<br>11 = 64 Dwords | R/W | 0x0 |
| 5:4 | Upstream Transaction Maximum Prefetch Count | This register bits designates the maximum prefetch count for upstream Memory Read transactions.<br>Reset value: 0.<br>00 = 16 Dwords<br>01 = 32 Dwords<br>10 = 48 Dwords<br>11 = 64 Dwords | R/W | 0x0 |
| 3:1 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 0 | Non-posted Flush | When set to 1’b1 clears the non-posted read completions from the primary interface, in the non-posted buffer, when a posted write is received from primary interface. The read request retry from the secondary device will be registered as a new non-posted read request.<br>When set to 0x0, the bridge follows PCI Bridge ordering rules. | R/W | 0x0 |

### 11.5.6 Secondary Bus Arbiter Preemption Control Register — Offset 0x4C

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 31:28 | Secondary bus arbiter preemption control | Controls the number of clock cycles after frame is asserted before preemption is enabled.<br>1xxx: Preemption off<br>0000 = Preemption enabled after 3 clock cycles after FRAME asserted<br>0001 = Preemption enabled after 3 clock cycle after FRAME asserted<br>0010 = Preemption enabled after 3 clock cycles after FRAME asserted<br>0011 = Preemption enabled after 4 clock cycle after FRAME asserted<br>0100 = Preemption enabled after 8 clock cycles after FRAME asserted<br>0101 = Preemption enabled after 16 clock cycles after FRAME asserted<br>0110 = Preemption enabled after 32 clock cycles after FRAME asserted<br>0111 = Preemption enabled after 64 clock cycles after FRAME asserted | R/W | 0x0 |

### 11.5.7 P_SERR_b Event Disable Register—Offset 0x64

This section describes the P_SERR_b event disable register.

Byte enable P_CBE_b<3:0> = xxx0b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 0 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |
| 1 | Posted write parity error | Controls the Tsi340's ability to assert P_SERR_b when a data parity error is detected on the target bus during a posted write transaction.<br>0 = Signal P_SERR_b is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>1 = Signal P_SERR_b is not asserted if this event occurs. | R/W | 0x0 |
| 2 | Posted write non delivery | Controls the Tsi340's ability to assert P_SERR_b when it is unable to deliver posted write data after 2^24 attempts.<br>0 = Signal P_SERR_b is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>1 = Signal P_SERR_b is not asserted if this event occurs. | R/W | 0x0 |
| 3 | Target abort during posted write | Controls the Tsi340's ability to assert P_SERR_b when it receives a target abort when attempting to deliver posted write data.<br>0 = Signal P_SERR_b is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>1 = Signal P_SERR_b is not asserted if this event occurs. | R/W | 0x0 |
| 4 | Master abort on posted write | Controls the Tsi340's ability to assert P_SERR_b when it receives a master abort when attempting to deliver posted write data.<br>0 = Signal P_SERR_b is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>1 = Signal P_SERR_b is not asserted if this event occurs. | R/W | 0x0 |
| 5 | Delayed write nondelivery | Controls the Tsi340's ability to assert P_SERR_b when it is unable to deliver delayed write data after 2^24 attempts.<br>0 = Signal P_SERR_b is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>1 = Signal P_SERR_b is not asserted if this event occurs. | R/W | 0x0 |
| 6 | Delayed read—no data from target | Controls the Tsi340's ability to assert P_SERR_b when it is unable to transfer any read data from the target after 2^24 attempts.<br>0 = Signal P_SERR_b is asserted if this event occurs and the SERR# enable bit in the command register is set.<br>1 = Signal P_SERR_b is not asserted if this event occurs. | R/W | 0x0 |
| 7 | Reserved | Reserved. Returns 0 when read. | R | 0x0 |

### 11.5.8 Secondary Clock Control Register—Offset 0x68

This section describes the secondary clock control register.

Byte enable P_CBE_b<3:0> = xx00b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 1:0 | S_CLKOUT0 disable | 00, 01, 10 = S_CLKOUT0 enabled (00 is the default).<br>11 = S_CLKOUT0 disabled and driven high. | R/W | 0x0 |
| 3:2 | S_CLKOUT1 disable | 00, 01, 10 = S_CLKOUT1 enabled (00 is the default).<br>11 = S_CLKOUT1 disabled and driven high. | R/W | 0x0 |
| 5:4 | S_CLKOUT2 disable | 00, 01, 10 = S_CLKOUT2 enabled (00 is the default).<br>11 = S_CLKOUT2 disabled and driven high. | R/W | 0x0 |
| 7:6 | S_CLKOUT3 disable | 00, 01, 10 = S_CLKOUT3 enabled (00 is the default).<br>11 = S_CLKOUT3 disabled and driven high. | R/W | 0x0 |
| 8 | Reserved | Reserved: Return 0 when read. | R | 0x0 |
| 13:9 | Reserved | Reserved. Reset to 0x1F | R | 0x1F |
| 15:14 | Reserved | Reserved: Return 0 when read. | R | 0x0 |

### 11.5.9 P_SERR_b Status Register — Offset 0x68

This status register indicates the reason for Tsi340's assertion of P_SERR_b.

Byte enable P_CBE_b<3:0> = x0xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 16 | Address parity error | 1 = Signal P_SERR_b was asserted because an address parity error was detected on either the primary or secondary PCI bus. | R/W1TC | 0x0 |
| 17 | Posted write data parity error | 1 = Signal P_SERR_b was asserted because a posted write data parity error was detected on the target bus. | R/W1TC | 0x0 |
| 18 | Posted write nondelivery | 1 = Signal P_SERR_b was asserted because the Tsi340 was unable to deliver posted write data to the target after 2^24 attempts. | R/W1TC | 0x0 |
| 19 | Target abort during posted write | 1 = Signal P_SERR_b was asserted because the Tsi340 received a target abort when delivering posted write data. | R/W1TC | 0x0 |
| 20 | Master abort during posted write | 1 = Signal P_SERR_b was asserted because the Tsi340 received a master abort when attempting to deliver posted write data | R/W1TC | 0x0 |
| 21 | Delayed write nondelivery | 1 = Signal P_SERR_b was asserted because the Tsi340 was unable to deliver delayed write data after 2^24attempts. | R/W1TC | 0x0 |
| 22 | Delayed read—no data from target | 1 = Signal P_SERR_b was asserted because the Tsi340 was unable to read any data from the target after 2^24 attempts. | R/W1TC | 0x0 |
| 23 | Delayed transaction master timeout | 1 = Signal P_SERR_b was asserted because a master did not repeat a read or write transaction before the master timeout counter expired on the initiator's PCI bus. | R/W1TC | 0x0 |

### 11.5.10 CLKRUN Register — Offset 0x6C

Byte enableP_CBE_b<3:0> = 0xxxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 24 | Secondary Clock Stop Status | 0 = Secondary clock not stopped<br>1 = Secondary clock stopped. | R | 0x0 |
| 25 | Secondary CLKRUN Enable | 0 = Disable secondary CLKRUN<br>1 = Enable secondary CLKRUN | R/W | 0x0 |
| 26 | Primary Clock Stop | 0 = Allow primary clock to stop if secondary clock is stopped<br>1 = Always keep primary clock running | R/W | 0x0 |
| 27 | Primary CLKRUN Enable | 0 = Disable primary CLKRUN<br>1 = Enable primary CLKRUN | R/W | 0x0 |
| 28 | CLKRUN mode | 0 = Stop the secondary clock only on request from the primary bus<br>1 = Stop the secondary clock whenever the secondary bus is idle and there are no requests from the primary bus | R/W | 0x0 |
| 31:29 | Reserved | Reserved. Reset to 0. | R | 0x0 |

### 11.5.11 Port Option Register — Offset 0x74

Byte enableP_CBE_b<3:0> = xx00b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 0 | Reserved | Reserved. Reset to 0 | R | 0x0 |
| 1 | Primary Memory Read Command Alias Enable | Controls bridge detection mechanism for matching memory read retry cycles from the initiator on the primary interface<br>0 = exact matching memory read retry cycles from initiator on the primary<br>interface<br>1 = alias MEMRL or MEMRM to MEMR for memory read retry cycles from the initiator on the primary interface | R/W | 0x1 |
| 2 | Reserved | Reserved. Reset to 0 | R | 0x0 |
| 3 | Secondary Memory Read Command Alias Enable | Controls bridge detection mechanism for matching memory read retry cycles from the initiator on the secondary interface<br>0 = exact matching memory read retry cycles from initiator on the secondary interface<br>1 = alias MEMRL or MEMRM to MEMR for memory read retry cycles from the initiator on the secondary interface | R/W | 0x1 |
| 4 | Reserved | Reserved. Reset to 0 | R | 0x0 |
| 5 | Primary Memory Read<br>Line/Multiple Alias Enable | Control bridge detection mechanism for matching memory read line/multiple cycles from the initiator on the primary interface<br>0 = exact matching for memory read line/multiple retry cycles from the initiator on the primary interface<br>1 = alias MEMRL to MEMRM or MEMRM to MEMRL for memory read retry cycles from the initiator on the primary interface | R/W | 0x1 |
| 6 | Secondary Memory Read Line/Multiple Alias Enable | Control bridge detection mechanism for matching memory read line/multiple cycles from the initiator on the secondary interface<br>0 = exact matching for memory read line/multiple retry cycles from the initiator on the secondary interface<br>1 = alias MEMRL to MEMRM or MEMRM to MEMRL for memory read retry cycles from the initiator on the secondary interface | R/W | 0x1 |

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 7 | Primary Memory Write and Invalidate Command Alias Disable | Controls bridge detection mechanism for matching posted memory write and invalidate cycles from the initiator on the primary interface<br>0 = When accepting MEMWI command at the primary interface, bridge converts MEMWI to MEMW command on the destination interface<br>1 = When accepting MEMWI command at the primary interface, bridge does not convert MEMWI to MEMW command on the destination interface | R/W | 0x0 |
| 8 | Secondary Memory Write and Invalidate Command Alias Disable | Controls bridge detection mechanism for matching posted memory write and invalidate cycles from the initiator on the secondary interface<br>0 = When accepting MEMWI command at the secondary interface, bridge converts MEMWI to MEMW command on the destination interface<br>1 = When accepting MEMWI command at the secondary interface, bridge does not convert MEMWI to MEMW command on the destination interface | R/W | 0x0 |
| 15:9 | Reserved | Reserved. Returns 0 when read. Reset to 0 | R | 0x0 |

### 11.5.12 Capability ID Register—Offset 0x80

This section describes the capability ID register.

Byte enable P_CBE_b<3:0> = xxx0b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 7:0 | CAP_ID | Enhanced capabilities ID.<br>Indicates that these are power management enhanced capability registers. | R | 0x01 |

### 11.5.13 Next Item Pointer Register—Offset 0x80

This section describes the next item pointer register.

Byte enable P_CBE_b<3:0> = xx0xb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 15:8 | NEXT_ITEM | Indicates that Tsi340 supports more than one extended capability. | R | 0x90 |

### 11.5.14 Power Management Capabilities Register—Offset 0x80

This section describes the power management capabilities register.

Byte enable P_CBE_b<3:0> = 00xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 18:16 | PM_VER | Power Management Revision.<br>Indicates that this device is compliant with *Revision 1.1 of the PCI Power Management Interface Specification*. | R | 0x10 |
| 19 | PME#Clock | PME# Clock Required.<br>Reads as 0 to indicate that this device does not support the PME# pin. | R | 0x0 |
| 20 | AUX | Auxiliary Power Support.<br>Reads as 0 to indicate that this device does not have PME# support or an auxiliary power source. | R | 0x0 |
| 21 | DSI | Device Specific Initialization. Reads as 0 to indicate that this device does not have device specific initialization requirements. | R | 0x0 |
| 24:22 | Reserved | Reserved. | R | 0x0 |
| 25 | D1 | D1 Power State Support.<br>Reads as 0 to indicate that this device does not support the D1 power management state. | R | 0x0 |
| 26 | D2 | D2 Power State Support.<br>Reads as 0 to indicate that this device does not support the D2 power management state. | R | 0x0 |
| 31:27 | PME_SUP | PME# Support.<br>Reads as 0 to indicate that this device does not support the PME# pin. | R | 0x0 |

### 11.5.15 Power Management Data Register—Offset 0x84

This section describes the power management control and status register.

Byte enable P_CBE_b<3:0> = xx00b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 1:0 | PWR_STATE | Power State.<br>Reflects the current power state of this device. If an unimplemented power state is written to this register, Tsi340 completes the write transaction, ignores the write data, and does not change the value of this field. Writing a value of D0 when the previous state was D3 causes a chip reset to occur (without asserting S_RST_b).<br>00 = D0<br>01 = D1 (not implemented)<br>10 = D2 (not implemented)<br>11 = D3. | R/W | 0x0 |
| 7:2 | Reserved | Reserved. Reads as 000000b. | R | 0x0 |
| 8 | PME_EN | PME# Enable.<br>Reads as 0x0 because the PME# pin is not implemented. | R | 0x0 |
| 12:9 | DATA_SEL | Data Select.<br>Reads as 0x0 because the data register is not implemented. | R | 0x0 |
| 14:13 | DATA_SCALE | Data Scale.<br>Reads as 00b because the data register is not implemented. | R | 0x0 |
| 15 | PME_STAT | PME Status.<br>Reads as 0x0 because the PME# pin is not implemented. | R | 0x0 |

### 11.5.16 PMCSR_BSE Register — Offset 0x84

Byte enableP_CBE_b<3:0> = x0xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 21:16 | Reserved | Reserved. Read only as 0x000000 | R | 0x0 |
| 22 | B2_B3 | B2_B3 Support for D3hot.<br>This bit is not defined and reads as 0. | R | 0x0 |
| 23 | BPCC_EN | Bus Power/Clock Control Enable.<br>Reads as 0x0 because the BPCC_EN is not implemented. | R | 0x0 |

### 11.5.17 HS Capability ID Register — Offset 0x90

Byte enableP_CBE_b<3:0> = xxx0b

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 7:0 | Hot Swap Capability ID | Indicates that this register set of the capability list is a Hot Swap Register Set. | R | 0x6 |

### 11.5.18 HS Next Item Pointer Register — Offset 0x90

Byte enableP_CBE_b<3:0> = xx0xb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 15:8 | Hot Swap Next Pointer | Indicates there are more list items in the capabilities list. | R | 0 |

### 11.5.19 HS Control Status Register — Offset 0x90

Byte enableP_CBE_b<3:0> = x0xxb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 16 | DHA | Device Hiding Arm<br>When this bit has a value of 1'b1, the Device Hiding will be armed. The device hiding is useful in hiding the device from software during board removal. Writing a 1'b1 to this bit can arm device hiding. The hardware sets this bit after P_RST# is de-asserted and switch handle is open. The hardware automatically clears this bit after it enters INS state. | R/W | 0x0 |

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 17 | EIM | ENUM Interrupt Mask<br>This bit allows the ENUM# pin to be masked by software. Writing a logical 1'b1 to this bit masks the ENUM# pin from being driven. Writing a logical 1'b0 to this bit will enable ENUM# to be driven. | R/W | 0x0 |
| 18 | PIE | Pending INS/EXT<br>Returns 0x0 when read. | R | 0x0 |
| 19 | LOO | LED On Off<br>This bit controls an external LED indicator for user feedback. When software writes a logical one to this register, the LED is illuminated. When a logical zero is written to this bit the LED is not illuminated. | R/W | 0x0 |
| 21:20 | PI | Programming Interface<br>Returns 0x0 when read. | R | 0x0 |
| 22 | EXT | Extraction<br>The bridge sets this bit to indicate software that a board is about to be removed from the system. The bridge asserts ENUM# when this bit is set. | R/W1TC | 0x0 |
| 23 | INS | Insertion<br>The bridge sets this bit to indicate software that a board has been freshly inserted. This bit will be set only after the following events:<br>• Ejector Handle is closed<br>• P_RST# deasserted<br>The bridge asserts ENUM# when this bit is set. | R/W1TC | 0x0 |

### 11.5.20 Miscellaneous Control Register — Offset 0xC0

Byte enableP_CBE_b<3:0> = xx0xb

| Bit | Name | Description | Type | Reset Value |
|---|---|---|---|---|
| 8 | Legacy ISA I/O Enable | 0 = The I/O addresses will not be claimed by the bridge and will not be forwarded on to the secondary bus<br>1 = The following I/O addresses will be forwarded on to the secondary bus<br>• Game port: 0x0200 - 0x0207<br>• FM: 0x0388 - 0x038B<br>• Audio: 0x0220 - 0x0233<br>• MIDI: 0x0330 - 0x0331 | R/W | 0x0 |
| 15:9 | Reserved | Returns 0 when read | R | 0 |

# 12. Packaging

This chapter discusses the following topics about Tsi340's packaging:



## 12.1 Mechanical Diagram

The 128-pin PQFP package figures have symbols that describe measurements of the package. The following table outlines the symbol values.

**Table 25: PQFP Symbol Values**

| Symbol | Millimeter | | |
|---|---|---|---|
| | Minimum | Nominal | Maximum |
| A | - | - | 3.40 |
| A1 | 0.25 | - | - |
| A2 | 2.50 | 2.72 | 2.90 |
| D | 23.20 BASIC | | |
| D1 | 20.00 BASIC | | |
| E | 17.20 BASIC | | |
| E1 | 14.00 BASIC | | |
| R2 | 0.13 | - | 0.30 |
| R1 | 0.13 | - | - |
| Theta | 0 | - | 7 |
| Theta1 | 0 | - | - |
| Theta2 Theta3 | 15 REF | | |
| c | 0.11 | 0.15 | 0.23 |
| L | 0.73 | 0.88 | 1.03 |
| L1 | 1.60 REF | | |

**Table 25: PQFP Symbol Values**

| Symbol | Millimeter | | |
|---|---|---|---|
| | Minimum | Nominal | Maximum |
| S | 0.20 | - | - |
| b | 0.17 | 0.20 | 0.27 |
| e | 0.50 BASIC | | |
| D2 | 18.50 | | |
| E2 | 12.50 | | |
| aaa | 0.20 | | |
| bbb | 0.20 | | |
| ccc | 0.08 | | |
| ddd | 0.08 | | |

The following figures show the Tsi340 mechanical diagram.

**Figure 13: Tsi340 PQFP Package - Top View**

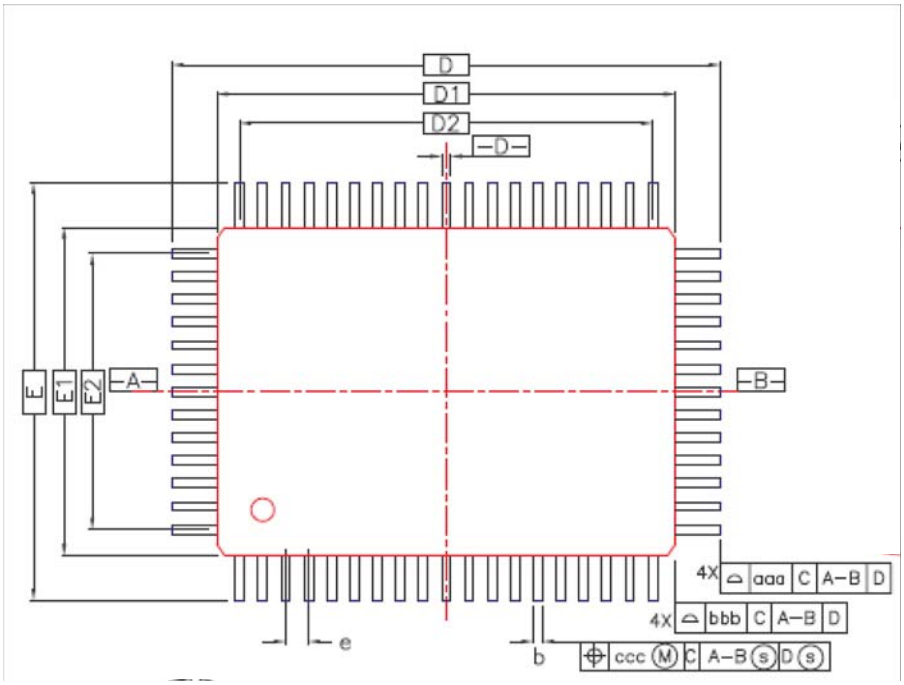


**Figure 14: Tsi340 PQFP Package - Side View**

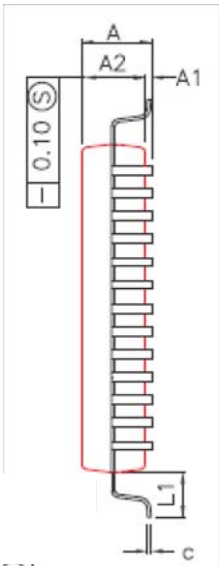



**Figure 15: Tsi340 PQFP Package - Side View**

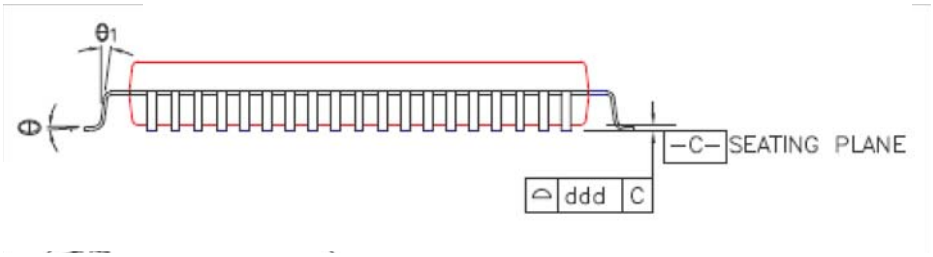
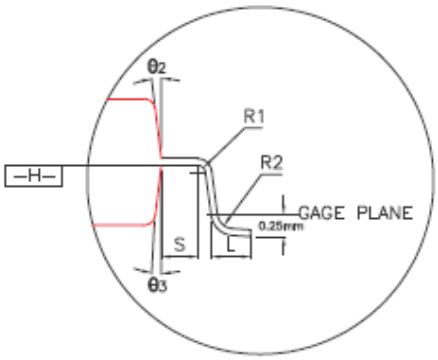



**Figure 16: Tsi340 PQFP Package - Side View**



## 12.2 Thermal Characteristics

Heat generated by the packaged IC has to be removed from the package to ensure that the IC is maintained within its functional and maximum design temperature limits. If heat buildup becomes excessive, the IC temperature may exceed the temperature limits. A consequence of this is that the IC may fail to meet the performance specifications and the reliability objectives may be affected.

Failure mechanisms and failure rate of a device have an exponential dependence of the IC operating temperatures. Thus, the control of the package temperature, and by extension the Junction Temperature, is essential to ensure product reliability. The Tsi340 is specified safe for operation when the Junction Temperature is within the recommended limits.

Table 26 shows the simulated Theta jb and Theta jc thermal characteristics of the Tsi340 packages.

**Table 26: Thermal Characteristics of the Tsi340**

| Interface | Tsi340 PQFP Result |
|---|---|
| $Theta_{jb}$ (junction to board) | 24.8 °C/watt |
| $Theta_{jc}$ (junction to case) | 13.1 °C/watt |

### 12.2.1 Junction-to-Ambient Thermal Characteristics (Theta ja)

Table 27 shows the simulated $Theta_{ja}$ thermal characteristic of the Tsi340 PQFP package. The results in Table 27 are based on a JEDEC Thermal Test Board configuration (JESD51-9) and do not factor in system level characteristics. As such, these values are for reference only.

The $Theta_{ja}$ thermal resistance characteristics of a package depend on multiple system level variables (see “System-level Characteristics” on page 145).

**Table 27: Simulated Junction to Ambient Characteristics**

| Package | $Theta_{ja}$ at specified airflow (no Heat Sink) | | |
|---|---|---|---|
| | 0 m/s | 1 m/s | 2 m/s |
| Tsi340 PQFP | 29.5 °C/watt | 27.2 °C/watt | 26.2 °C/watt |

### 12.2.2 System-level Characteristics

The thermal resistance characteristics of a package depend on multiple variables other than the package. In an application, the following system-level characteristics and environmental issues must be taken into account:

- Package mounting (vertical / horizontal)
- System airflow conditions (laminar / turbulent)
- Heat sink design and thermal characteristics
- Heat sink attachment method
- PWB size, layer count and conductor thickness
- Influence of the heat dissipating components assembled on the PWB (neighboring effects)

### 0.0.1 Example on Thermal Data Usage

Based on the $Theta_{JA}$ data and specified conditions, the following formula can be used to derive the junction temperature (Tj) of the Tsi340 with a 0m/s airflow:

- $Tj = Theta_{JA} * P + Tamb.$

  Where: Tj is Junction Temperature, P is the Power consumption, Tamb is the Ambient Temperature

Assuming a power consumption (P) of 0.75 W and an ambient temperature (Tamb) of 85°C, the resulting junction temperature (Tj) for the PQFP package would be 107°C.

## 12.3 Moisture Sensitivity

The Tsi340’s moisture sensitivity level (MSL) is three.

# 13. Ordering Information

Topics discussed include the following:



## 13.1 Part Numbers

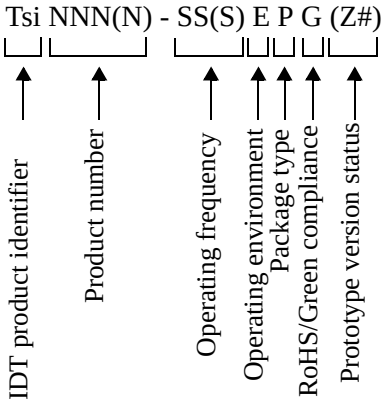
The following table contains ordering information for the Tsi340.

**Table 28: Part Numbers**

| Part Number | Frequency | Temperature | Package | Pin Count |
|---|---|---|---|---|
| Tsi340-66CQY | 66 MHz | Extended Commercial (0 to 85) | RoHS | 128 |

## 13.2 Part Numbering Information

The part numbering system is explained as follows.




- ( ) – Indicates optional characters.
- Tsi – IDT system interconnect product identifier.
- NNNN – Product number (may be three or four digits).
- SS(S) – Maximum operating frequency or data transfer rate of the fastest interface. For operating frequency numbers, M and G represent MHz and GHz. For transfer rate numbers, M and G represent Mbps and Gbps.

- E – Operating environment in which the product is guaranteed. This code may be one of the following characters:
  - C - Commercial temperature range (0 to +70°C)
  - I - Industrial temperature range (-40 to +85°C)
  - E - Extended temperature range (-55 to +125°C)
- P – The Package type of the product:
  - B - Ceramic ball grid array (CBGA)
  - E, L, J, and K - Plastic ball grid array (PBGA)
  - G - Ceramic pin grid array (CPGA)
  - M - Small outline integrated circuit (SOIC)
  - Q - Plastic quad flatpack (QFP)
- G – IDT products fit into three RoHS-compliance categories:
  - Y - RoHS Compliant (6of6) – These products contain none of the six restricted substances above the limits set in the EU Directive 2002/95/EC.
  - Y - RoHS Compliant (Flip Chip) – These products contain only one of the six restricted substances: Lead (Pb). These flip-chip products are RoHS compliant through the Lead exemption for Flip Chip technology, Commission Decision 2005/747/EC, which allows Lead in solders to complete a viable electrical connection between semiconductor die and carrier within integrated circuit Flip Chip packages.
  - V - RoHS Compliant/Green - These products follow the above definitions for RoHS Compliance and meet JIG (Joint Industry Guide) Level B requirements for Brominated Flame Retardants (other than PBBs and PBDEs).
- Z# – Prototype version status (optional). If a product is released as a prototype then a “Z” is added to the end of the part number. Further revisions to the prototype prior to production release would add a sequential numeric digit. For example, the first prototype version of device would have a “Z,” a second version would have “Z1,” and so on. The prototype version code is dropped once the product reaches production status.

## IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers skilled in the art designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only for development of an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising out of your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use o any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.0 Mar 2020)


### Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan

www.renesas.com

### Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit:

www.renesas.com/contact/


### Trademarks

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.