PCI 6520CB Data Book

Preliminary

PCI 6520CB Data Book

Version 2.0

May 2004

Website: http://www.plxtech.com
Technical Support: http://www.plxtech.com/support/
Phone: 408 774-9060
800 759-3735
FAX: 408 774-2169

Preliminary

Order Number: PCI 6520CB-SIL-DB-P1-2.0

Printed in the USA, May 2004

CONTENTS

Figures . . . . . **xi**

Tables . . . . . **xiii**

Registers . . . . . **xv**

Preface . . . . . **xvii**

Supplemental Documentation xvii
Data Assignment Conventions xviii
Revision History xviii

Feature Summary . . . . . **xix**

1. Introduction **1-1**

1.1. Company and Product Information 1-1
1.2. FastLane PCI 6000 Bridge Series 1-1
1.2.1. PCI 6520 1-5
1.3. Feature Description 1-5
1.4. Applications 1-6
1.4.1. Multiple Device Expansion 1-6

2. Functional Overview **2-1**

2.1. General Operation 2-1
2.2. Write Transactions 2-2
2.3. Read Transactions 2-2

3. Pin Description **3-1**

3.1. Pin Summary 3-1
3.2. Pull-Up and Pull-Down Resistor Recommendations 3-2
3.2.1. PCI Bus Interface Pins 3-2
3.2.2. Clock-Related Pins 3-3
3.2.3. Reset Pins 3-3
3.2.4. JTAG/Boundary Scan Pins 3-3
3.2.5. Serial EEPROM Pins 3-3
3.2.6. GPIO Pins 3-3
3.2.7. Miscellaneous Pins 3-3
3.2.8. System Voltage Pins 3-4
3.3. Power Supply De-Coupling 3-5
3.4. Pinout 3-6

4. Clocking **4-1**

4.1. Primary and Secondary Clock Inputs 4-1
4.2. Secondary Clock Outputs 4-1
4.2.1. Disabling Secondary Clock Outputs 4-1
4.2.2. Secondary Clock Control 4-1
4.3. Using an External Clock Source 4-4
4.4. Frequency Division Options 4-4
4.5. Running Secondary Port Faster than Primary Port 4-4
4.6. PLL and Clock Jitter 4-4

4.7. Detecting PCI Bus Speed with the Reference Clock 4-6
4.8. Primary or Secondary Clock Frequency Measurement 4-6

5. Reset and Initialization 5-1

5.1. PCI-XCAP Connections and Operating Frequency 5-1
5.1.1. Primary Port PCI-XCAP Connection 5-1
5.1.2. Secondary Port PCI-XCAP Connection 5-1
5.2. Secondary Bus Mode and Frequency Initialization Sequence 5-1
5.3. Conventional PCI Mode 66 MHZ Operation 5-2
5.4. Reset 5-2
5.4.1. Power Good Reset 5-2
5.4.2. Primary Reset Input 5-3
5.4.3. Secondary Reset Output 5-3
5.4.4. JTAG Reset 5-4
5.4.5. Software Resets 5-4
5.4.6. Power Management Internal Reset 5-4
5.4.7. Reset Inputs Effect on PCI 6520 5-4
5.4.8. Pin States during PWRGD and Primary Reset 5-5
5.5. Register Initialization 5-7
5.5.1. Default Initialization 5-7
5.5.2. Serial EEPROM Initialization 5-7
5.5.3. Host Initialization 5-7

6. Registers 6-1

6.1. PCI Configuration Register Address Mapping 6-2
6.1.1. PCI Type 1 Header 6-4
6.1.2. Device-Specific 6-16
6.1.2.1. Chip, Diagnostic, and Arbiter Control 6-16
6.1.2.2. Primary Flow-Through Control 6-18
6.1.2.3. Timeout Control 6-19
6.1.2.4. Miscellaneous Options 6-20
6.1.2.5. Prefetch Control 6-22
6.1.2.6. Secondary Flow-Through Control 6-26
6.1.2.7. Buffer and Internal Arbiter Control 6-27
6.1.2.8. Test and Serial EEPROM 6-29
6.1.2.9. Timer 6-31
6.1.2.10. Primary System Error Event 6-32
6.1.2.11. GPIO[3:0] 6-33
6.1.2.12. Clock Control 6-34
6.1.2.13. Primary System Error Status 6-35
6.1.2.14. Clock Run 6-36
6.1.2.15. Private Memory 6-37
6.1.2.16. Read-Only Register Control 6-38
6.1.2.17. GPIO[7:4] 6-39
6.1.2.18. Extended and Smart Prefetch 6-40
6.1.2.19. Power Management Capability 6-46
6.1.2.20. VPD Capability 6-49
6.1.2.21. PCI-X Capability 6-50

7. Serial EEPROM 7-1

7.1. Overview 7-1
7.2. Serial EEPROM Access 7-1
7.3. Serial EEPROM Autoload Mode at Reset 7-1
7.4. Serial EEPROM Data Structure 7-2
7.4.1. Serial EEPROM Address and Corresponding PCI 6520 Registers 7-3

8. PCI Bus Operation 8-1

8.1. Conventional PCI Transactions 8-1
8.2. Single Address Phase 8-2
8.3. Dual Address Phase 8-2
8.4. Device Select (DEVSEL#) Generation 8-2
8.5. Data Phase 8-2
8.5.1. Posted Write Transactions 8-3
8.5.2. Memory Write and Invalidate Transactions 8-3
8.5.3. Delayed Write Transactions 8-3
8.5.4. Write Transaction Address Boundaries 8-5
8.5.5. Buffering Multiple Write Transactions 8-5
8.5.6. Read Transactions 8-5
8.5.7. Prefetchable Read Transactions 8-6
8.5.8. Non-Prefetchable Read Transactions 8-6
8.5.9. Read Prefetch Address Boundaries 8-6
8.5.10. Delayed Read Requests 8-7
8.5.11. Delayed Read Completion with Target 8-7
8.5.12. Delayed Read Completion on Initiator Bus 8-7
8.5.13. Configuration Transactions 8-8
8.5.14. PCI 6520 Type 0 Access 8-8
8.5.15. Type 1-to-Type 0 Translation 8-8
8.5.16. Type 1-to-Type 1 Forwarding 8-11
8.5.17. Special Cycles 8-11
8.6. Conventional PCI Mode Transaction Termination 8-12
8.6.1. PCI 6520-Initiated Master Termination 8-12
8.6.2. Master Abort Received by PCI 6520 8-13
8.6.3. Target Termination Received by PCI 6520 8-13
8.6.3.1. Posted Write Target Termination Response 8-14
8.6.3.2. Delayed Write Target Termination Response 8-15
8.6.3.3. Delayed Read Target Termination Response 8-16
8.6.4. PCI 6520-Initiated Target Termination 8-17
8.6.4.1. Target Retry 8-17
8.6.4.2. Target Disconnect 8-18
8.6.4.3. Target Abort 8-18

9. PCI-X Bus Operation 9-1

9.1. Overview 9-1
9.2. General Bus Rules 9-1
9.2.1. Initiator Rules 9-3
9.2.2. Target Rules 9-3
9.2.3. Bus Arbitration Rules 9-3
9.2.4. Configuration Transaction Rules 9-4
9.2.5. Parity Error Rules 9-4
9.2.6. Bus Data Width Rules 9-4
9.2.7. Split Transaction Rules 9-4
9.3. PCI-X Sequences 9-4
9.4. ADB and Buffer Size 9-5
9.5. Dependencies between AD and CBE Buses 9-5
9.6. PCI-X Command Encoding 9-6
9.7. Attributes 9-7
9.8. Burst Transactions 9-9
9.8.1. Burst Write and Split Completion 9-9
9.8.2. Burst Read Transactions 9-9
9.9. DWORD Transactions 9-9
9.10. Device Select Timing 9-10
9.11. Wait States and Target Initial Latency 9-10

9.12. Split Transactions . . . 9-11
9.12.1. Split Completion Transaction . . . 9-12
9.12.2. Immediate Completion by the Completer . . . 9-12
9.12.3. Split Completion Address . . . 9-12
9.12.4. Completer Attributes . . . 9-14
9.12.5. Split Completion Acceptance Requirement . . . 9-15
9.12.6. Split Completion Messages . . . 9-15
9.13. PCI-X Mode Transaction Termination . . . 9-18
9.13.1. PCI 6520 Initiator Termination . . . 9-18
9.13.1.1. Byte Count Disconnection or Satisfaction . . . 9-18
9.13.1.2. PCI 6520 Master Abort Termination . . . 9-18
9.13.2. PCI 6520 Target Termination . . . 9-18
9.13.2.1. PCI 6520 Disconnects at Next ADB . . . 9-20
9.13.2.2. PCI 6520 Retry Termination . . . 9-20
9.13.2.3. PCI 6520 Split Response Termination . . . 9-20
9.14. PCI-X Mode Bus and Data Transfer Width . . . 9-20
9.15. Connecting Conventional PCI and PCI-X Interfaces . . . 9-21
9.15.1. Conventional PCI Requester, PCI-X Completer . . . 9-21
9.15.2. PCI-X Requester, Conventional PCI Completer . . . 9-22

10. Address Decoding . . . 10-1

10.1. Overview . . . 10-1
10.2. Address Ranges . . . 10-1
10.2.1. I/O Address Decoding . . . 10-1
10.2.1.1. I/O Base and Limit Address Registers . . . 10-1
10.3. Memory Address Decoding . . . 10-2
10.3.1. Memory-Mapped I/O Base and Limit Address Registers . . . 10-3
10.3.2. Prefetchable Memory Base and Limit Address Registers . . . 10-3
10.4. ISA Mode . . . 10-4
10.5. VGA Support . . . 10-4
10.5.1. VGA Mode . . . 10-4
10.5.2. VGA Snoop Mode . . . 10-5
10.6. Private Device Support . . . 10-5

11. Transaction Ordering . . . 11-1

11.1. Conventional PCI Transaction Ordering . . . 11-1
11.1.1. Transactions Governed by Ordering Rules . . . 11-1
11.1.2. General Ordering Guidelines . . . 11-1
11.1.3. Ordering Rules . . . 11-2
11.1.4. Data Synchronization . . . 11-3
11.2. PCI-X Transaction Ordering . . . 11-4
11.2.1. Relaxed Ordering Attribute Bit . . . 11-4
11.2.2. Split Transactions . . . 11-4

12. Error Handling . . . 12-1

12.1. Overview . . . 12-1
12.2. Address Parity Errors . . . 12-1
12.3. Attribute Parity Errors— PCI-X Mode . . . 12-1
12.4. Data Parity Errors . . . 12-1
12.4.1. Configuration Write Transactions to Configuration Space . . . 12-2
12.4.2. Read Transactions . . . 12-2
12.4.3. Posted Write Transactions . . . 12-2
12.4.4. Delayed Write Transactions . . . 12-3
12.4.4.1. Conventional PCI Mode . . . 12-3
12.4.4.2. PCI-X Mode . . . 12-4
12.4.5. Split Completion—PCI-X Mode . . . 12-4

12.5. Data Parity Error Reporting Summary 12-4
12.6. System Error (P_SERR#) Reporting. 12-13

13. Exclusive Access 13-1

13.1. Concurrent Locks 13-1
13.2. Acquiring Exclusive Access across PCI 6520. 13-1
13.3. Ending Exclusive Access 13-2

14. PCI Bus Arbitration 14-1

14.1. Overview. 14-1
14.2. Primary PCI Bus Arbitration 14-1
14.3. Secondary PCI Bus Arbitration 14-1
14.3.1. Secondary Bus Arbitration Using Internal Arbiter 14-1
14.3.2. Rotating-Priority Scheme. 14-2
14.3.3. Fixed-Priority Scheme 14-2
14.3.4. Secondary Bus Arbitration Using External Arbiter 14-3
14.4. Arbitration Bus Parking 14-3
14.4.1. Software Controlled PCI 64-Bit Extension Signals Parking 14-3

15. GPIO Interface 15-1

15.1. GPIO Interface Pins 15-1
15.2. GPIO Control Registers 15-1
15.3. GPIO Serial Stream 15-1

16. Supported Commands. 16-1

16.1. Primary Interface Command Set. 16-1
16.2. Secondary Interface Command Set 16-3

17. Bridge Behavior 17-1

17.1. Bridge Actions for Various Cycle Types 17-1
17.2. Abnormal Termination (Master Abort, Initiated by Bridge Master) 17-2
17.3. Parity and Error Reporting 17-2

18. PCI Flow-Through Optimization 18-1

18.1. Overview. 18-1
18.2. Precautions when Using Non-Optimized PCI Master Devices 18-1
18.3. Posted Write Flow Through 18-1
18.4. Delayed Read Flow Through 18-2
18.5. Read Cycle Optimization 18-2
18.5.1. Primary and Secondary Initial Prefetch Count. 18-2
18.5.2. Primary and Secondary Incremental Prefetch Count. 18-3
18.5.3. Primary and Secondary Maximum Prefetch Count 18-3
18.6. Read Prefetch Boundaries 18-3

19. FIFO Architecture. 19-1

19.1. Overview. 19-1
19.2. Memory Writes 19-2
19.2.1. PCI-to-PCI-X Memory Writes. 19-2
19.2.2. PCI-X-to-PCI Memory Writes. 19-2
19.2.3. PCI-X-to-PCI-X Memory Writes 19-2

19.3. Memory Reads . . . 19-2
19.3.1. PCI-to-PCI-X Memory Reads. . . . 19-2
19.3.1.1. Prefetched Data Timeout Flushing . . . 19-3
19.3.1.2. Setting the Prefetch Count . . . 19-4
19.3.1.2.1. PCI Read from Conventional PCI Port . . . 19-4
19.3.1.2.2. PCI Read from PCI-X Port . . . 19-4
19.3.2. PCI-X-to-PCI Memory Reads. . . . 19-4
19.3.3. PCI-X-to-PCI-X Memory Reads . . . 19-4

20. Power Management . . . 20-1

20.1. Overview . . . 20-1
20.2. Power Management Transitions . . . 20-1

21. VPD . . . 21-1

22. Testability/Debug . . . 22-1

22.1. JTAG Interface . . . 22-1
22.1.1. IEEE 1149.1 Test Access Port . . . 22-1
22.1.2. JTAG Instructions . . . 22-1
22.1.3. JTAG Boundary Scan . . . 22-2
22.1.4. JTAG Reset Input TRST# . . . 22-2

23. Electrical Specs . . . 23-1

23.1. General Electrical Specifications . . . 23-1
23.2. PLL and Clock Jitter . . . 23-3
23.3. PCI/PCI-X Signal Timing Specification . . . 23-5

24. Mechanical Specs . . . 24-1

24.1. Mechanical Dimensions . . . 24-1
24.2. Physical Layout with Pinout . . . 24-4

A. Using PCI 6520. . . . A-1

A.1. Application . . . A-2

B. PCI-X Clock and Frequency Initialization Sequence . . . B-1

B.1. Bus Speed and Type Detection . . . B-1
B.2. Secondary Clock Outputs. . . . B-2
B.3. Internal Clock Divider . . . B-2

C. PCI 6520CB and PCI 6540CB Pin Comparison . . . C-1

D. General Information . . . D-1

D.1. Package Ordering . . . D-1
D.2. United States and International Representatives, and Distributors . . . D-2
D.3. Technical Support . . . D-2

Index . . . Index-1

FIGURES

1-1. FastLane PCI 6000 Bridge Series 1-4
1-2. PCI 6520 PCI-X-to-PCI-X Bridge 1-5
1-3. Multiple Device Expansion 1-6
2-1. PCI 6520 Block Diagram 2-1
3-1. Worst-Case Power Dissipation Example 3-4
4-1. GPIO Clock Mask Implementation on System Board Example 4-2
4-2. Clock Mask and Load Shift Timing 4-2
6-1. Sample Memory Map of Smart Prefetch Upstream Memory, Regions 1 through 4 6-40
7-1. Serial EEPROM Data Structure 7-2
9-1. Requester Attribute Bit Assignments 9-8
9-2. Split Completion Address 9-13
9-3. Completer Attribute Bit Assignments 9-14
9-4. Split Completion Message Attribute Bit Assignments 9-16
14-1. Secondary Bus Arbiter Example 14-2
19-1. PCI 6520 FIFO Architecture 19-1
23-1. PCI/PCI-X Signal Timing Specification 23-5
24-1. PCI 6520 Mechanical Dimensions 24-1
24-2. PCI 6520 Physical Layout with Pinout—Topside View (A1–A10 through Y1–Y10) 24-4
24-3. PCI 6520 Physical Layout with Pinout—Topside View (A11–A20 through Y11–Y20) 24-5

TABLES

1-1. FastLane PCI 6000 Series PCI and PCI-X Bridge Product Comparison 1-2
3-1. Pin Type Abbreviations 3-1
3-2. Generic PCI Bus Interface Pins that follow *PCI r2.3* and *PCI-X r1.0b* Layout Guidelines 3-2
3-3. Clock Pin Pull-Up/Pull-Down Resistor Requirements 3-3
3-4. Primary PCI/PCI-X Bus Interface Pins 3-6
3-5. Secondary PCI/PCI-X Bus Interface Pins 3-10
3-6. Clock-Related Pins 3-15
3-7. Reset Pins 3-18
3-8. JTAG/Boundary Scan Pins 3-19
3-9. Serial EEPROM Pins 3-19
3-10. General Purpose I/O Pins 3-20
3-11. Miscellaneous Pins 3-21
3-12. Power, Ground, and No Connect Pins 3-24
4-1. GPIO Shift Register Operation 4-3
4-2. GPIO Serial Data Format 4-3
4-3. PCI Clock Frequency Division Ratios 4-4
4-4. M66EN and PCI-XCAP Encoding 4-5
4-5. PLL and Clock Jitter Parameters 4-5
5-1. Reset Input Effect on PCI 6520 5-4
5-2. Pin States during PWRGD and P_RSTIN# 5-5
6-1. PCI Configuration Register Address Mapping 6-2
6-2. Extended Register Map—Offset from Extended Register Index 6-41
6-3. Secondary Clock Frequency Values (PCIXSSR[8:6]; PCI:F2h) 6-51
7-1. Serial EEPROM Address and Corresponding PCI 6520 Registers 7-3
8-1. Conventional PCI Transactions 8-1
8-2. Write Transaction Forwarding 8-2
8-3. Write Transaction Disconnect Address Boundaries 8-5
8-4. Read Transaction Prefetching 8-5
8-5. Read Prefetch Address Boundaries 8-6
8-6. Device Number to IDSEL S_AD Pin Mapping 8-10
8-7. P_SERR# Assertion Requirements in Response to Master Abort on Posted Write 8-13
8-8. Response to Posted Write Target Termination 8-14
8-9. P_SERR# Assertion Requirements in Response to Posted Write Parity Error 8-14
8-10. Response to Delayed Write Target Termination 8-15
8-11. P_SERR# Assertion Requirements in Response to Delayed Write 8-15
8-12. Response to Delayed Read Target Termination 8-16
8-13. P_SERR# Assertion Requirements in Response to Delayed Read 8-16
9-1. Transaction Phase Definitions 9-2
9-2. Byte Lane Assignments 9-5
9-3. PCI-X Command Encoding 9-6
9-4. Requester Attribute Bit Definitions 9-8
9-5. DEVSEL# Timing 9-10
9-6. Target Initial Latency 9-11
9-7. Split Completion Address Bit Definitions 9-13

9-8. Completer Attribute Bit Definitions 9-14
9-9. Split Completion Message Bit Definitions 9-16
9-10. PCI 6520 Error Message Indices 9-17
9-11. PCI 6520 Data Phase Signaling 9-19
9-12. Conventional PCI-to-PCI-X Command Translation 9-21
9-13. PCI-X-to-Conventional PCI Command Translation 9-22
11-1. Conventional PCI Transaction Ordering Summary 11-3
11-2. PCI-X Transaction Ordering and Deadlock-Avoidance Rules 11-5
11-3. PCI-X Split Transactions—Case-by-Case Discussion 11-6
12-1. Primary Interface Parity Error Detected Bit Status 12-6
12-2. Secondary Interface Parity Error Detected Bit Status 12-7
12-3. Primary Interface Data Parity Error Detected Bit Status 12-8
12-4. Secondary Interface Data Parity Error Detected Bit Status 12-9
12-5. P_PERR# Assertion 12-10
12-6. S_PERR# Assertion 12-11
12-7. P_SERR# or S_SERR# for Data Parity Error Assertion 12-12
15-1. GPIO Pin Alternate Functions 15-1
16-1. Primary Interface Supported Commands 16-1
16-2. Secondary Interface Supported Commands 16-3
17-1. Bridge Actions for Various Cycle Types 17-1
18-1. Reprogramming Prefetch Registers 18-1
19-1. Prefetched Data Timeout Flushing 19-3
20-1. States and Related Actions during Power Management Transitions 20-1
22-1. PCI 6520 JTAG IDCODE Value 22-1
22-2. JTAG Instructions (IEEE Standard 1149.1-1990) 22-1
23-1. Maximum Ratings 23-1
23-2. Functional Operating Range 23-1
23-3. DC Electrical Characteristics 23-2
23-4. M66EN and PCI-XCAP Encoding 23-3
23-5. PLL and Clock Jitter Parameters 23-4
23-6. 66 MHz PCI and 133 MHz PCI-X Signal Timing for Figure 23-1 23-5
24-1. PCI 6520 Mechanical Dimensions for Figure 24-1 Symbols (in Millimeters) 24-2
C-1. PCI 6520CB Versus PCI 6540CB Pin Assignment Comparison C-1
D-1. Available Package D-1

REGISTERS

6-1. (PCIIDR; PCI:00h) PCI Configuration ID 6-4
6-2. (PCICR; PCI:04h) Primary PCI Command 6-4
6-3. (PCISR; PCI:06h) Primary PCI Status 6-5
6-4. (PCIREV; PCI:08h) PCI Revision ID 6-6
6-5. (PCICCR; PCI:09h – 0Bh) PCI Class Code 6-6
6-6. (PCICLSR; PCI:0Ch) PCI Cache Line Size 6-6
6-7. (PCILTR; PCI:0Dh) Primary PCI Bus Latency Timer 6-6
6-8. (PCIHTR; PCI:0Eh) PCI Header Type 6-7
6-9. (PCIBISTR; PCI:0Fh) PCI Built-In Self-Test 6-7
6-10. (PCIPBNO; PCI:18h) PCI Primary Bus Number 6-8
6-11. (PCISBNO; PCI:19h) PCI Secondary Bus Number 6-8
6-12. (PCISUBNO; PCI:1Ah) PCI Subordinate Bus Number 6-8
6-13. (PCISLTR; PCI:1Bh) Secondary PCI Bus Latency Timer 6-8
6-14. (PCIIOBAR; PCI:1Ch) I/O Base 6-9
6-15. (PCIIOLMT; PCI:1Dh) I/O Limit 6-9
6-16. (PCISSR; PCI:1Eh) Secondary PCI Status 6-10
6-17. (PCIMBAR; PCI:20h) Memory Base 6-11
6-18. (PCIMLMT; PCI:22h) Memory Limit 6-11
6-19. (PCIPMBAR; PCI:24h) Prefetchable Memory Base 6-12
6-20. (PCIPMLMT; PCI:26h) Prefetchable Memory Limit 6-12
6-21. (PCIPMBARU32; PCI:28h) Prefetchable Memory Base Upper 32 Bits 6-13
6-22. (PCIPMLMTU32; PCI:2Ch) Prefetchable Memory Limit Upper 32 Bits 6-13
6-23. (PCIIOBARU16; PCI:30h) I/O Base Upper 16 Bits 6-13
6-24. (PCIIOLMTU16; PCI:32h) I/O Limit Upper 16 Bits 6-13
6-25. (CAP_PTR; PCI:34h) New Capability Pointer 6-14
6-26. (PCIIPR; PCI:3Dh) PCI Interrupt Pin 6-14
6-27. (BCNTRL; PCI:3Eh) Bridge Control 6-14
6-28. (CCNTRL; PCI:40h) Chip Control 6-16
6-29. (DCNTRL; PCI:41h) Diagnostic Control 6-17
6-30. (ACNTRL; PCI:42h) Arbiter Control 6-17
6-31. (PFTCR; PCI:44h) Primary Flow-Through Control 6-18
6-32. (TOCNTRL; PCI:45h) Timeout Control 6-19
6-33. (MSCOPT; PCI:46h) Miscellaneous Options 6-20
6-34. (PITLPCNT; PCI:48h) Primary Initial Prefetch Count 6-22
6-35. (SITLPCNT; PCI:49h) Secondary Initial Prefetch Count 6-23
6-36. (PINCPCNT; PCI:4Ah) Primary Incremental Prefetch Count 6-24
6-37. (SINCPCNT; PCI:4Bh) Secondary Incremental Prefetch Count 6-24
6-38. (PMAXPCNT; PCI:4Ch) Primary Maximum Prefetch Count 6-25
6-39. (SMAXPCNT; PCI:4Dh) Secondary Maximum Prefetch Count 6-25
6-40. (SFTCR; PCI:4Eh) Secondary Flow-Through Control 6-26
6-41. (BUFCR; PCI:4Fh) Buffer Control 6-27
6-42. (IACNTRL; PCI:50h) Internal Arbiter Control 6-28
6-43. (TEST; PCI:52h) Test 6-29
6-44. (EEPCNTRL; PCI:54h) Serial EEPROM Control 6-29

6-45. (EEPADDR; PCI:55h) Serial EEPROM Address 6-30
6-46. (EEPDATA; PCI:56h) Serial EEPROM Data 6-30
6-47. (TMRCNTRL; PCI:61h) Timer Control 6-31
6-48. (TMRCNT; PCI:62h) Timer Counter 6-31
6-49. (PSERRED; PCI:64h) P_SERR# Event Disable 6-32
6-50. (GPIOOD[3:0]; PCI:65h) GPIO[3:0] Output Data 6-33
6-51. (GPIOOE[3:0]; PCI:66h) GPIO[3:0] Output Enable 6-33
6-52. (GPIOID[3:0]; PCI:67h) GPIO[3:0] Input Data 6-33
6-53. (CLKCNTRL; PCI:68h) Clock Control 6-34
6-54. (PSERRSR; PCI:6Ah) P_SERR# Status 6-35
6-55. (CLKRUN; PCI:6Bh) Clock Run 6-36
6-56. (PVTMBAR; PCI:6Ch) Private Memory Base 6-37
6-57. (PVTMLMT; PCI:6Eh) Private Memory Limit 6-37
6-58. (PVTMBARU32; PCI:70h) Private Memory Base Upper 32 Bits 6-37
6-59. (PVTMLMTU32; PCI:74h) Private Memory Limit Upper 32 Bits 6-37
6-60. (RRC; PCI:9Ch) Read-Only Register Control 6-38
6-61. (GPIOOD[7:4]; PCI:9Dh) GPIO[7:4] Output Data 6-39
6-62. (GPIOOE[7:4]; PCI:9Eh) GPIO[7:4] Output Enable 6-39
6-63. (GPIOID[7:4]; PCI:9Fh) GPIO[7:4] Input Data 6-39
6-64. (EXTRIDX; PCI:D3h) Extended Register Index 6-41
6-65. (EXTRDATA; PCI:D4h) Extended Register Data 6-41
6-66. (SPUL32BAR1; EXT:10h) Region 1 Upstream Lower 32-Bit Smart Prefetch BAR 6-41
6-67. (SPUU32BAR1; EXT:11h) Region 1 Upstream Lower 32-Bit Smart Prefetch BAR 6-42
6-68. (SPUL32BAR2; EXT:12h) Region 2 Upstream Lower 32-Bit Smart Prefetch BAR 6-42
6-69. (SPUU32BAR2; EXT:13h) Region 2 Upstream Lower 32-Bit Smart Prefetch BAR 6-42
6-70. (SPUL32BAR3; EXT:14h) Region 3 Upstream Lower 32-Bit Smart Prefetch BAR 6-43
6-71. (SPUU32BAR3; EXT:15h) Region 3 Upstream Lower 32-Bit Smart Prefetch BAR 6-43
6-72. (SPUBARD*x*; EXT:16h–19h) Regions 1–4 Upstream Smart Prefetch BAR Descriptors 6-43
6-73. (PMCAPID; PCI:DCh) Power Management Capability ID 6-46
6-74. (PMNEXT; PCI:DDh) Power Management Next Capability Pointer 6-46
6-75. (PMC; PCI:DEh) Power Management Capabilities 6-47
6-76. (PMCSR; PCI:E0h) Power Management Control/Status 6-48
6-77. (PMCSR_BSE; PCI:E2h) PMCSR Bridge Supports Extensions 6-48
6-78. (PMCDATA; PCI:E3h) Power Management Data 6-48
6-79. (PVPDID; PCI:E8h) Vital Product Data Capability ID 6-49
6-80. (PVPD_NEXT; PCI:E9h) Vital Product Data Next Capability Pointer 6-49
6-81. (PVPDAD; PCI:EAh) Vital Product Data Address 6-49
6-82. (PVPDATA; PCI:ECh) VPD Data 6-49
6-83. (PCIXCAPID; PCI:F0h) PCI-X Capability ID 6-50
6-84. (PCIX_NEXT; PCI:F1h) PCI-X Next Capability Pointer 6-50
6-85. (PCIXSSR; PCI:F2h) PCI-X Secondary Status 6-50
6-86. (PCIXBSR; PCI:F4h) PCI-X Bridge Status 6-52
6-87. (PCIXUPSTR; PCI:F8h) PCI-X Upstream Split Transaction 6-53
6-88. (PCIXDNSTR; PCI:FCh) PCI-X Downstream Split Transaction 6-53

PREFACE

The information contained in this document is subject to change without notice. Although an effort has been made maintain accurate information, there may be misleading or even incorrect statements made herein.

Supplemental Documentation

The following is a list of documentation to provide further details:

- *PCI Local Bus Specification, Revision 2.1*, June 1, 1995
  PCI Special Interest Group (PCI-SIG)
  5440 SW Westgate Drive #217, Portland, OR 97221 USA
  Tel: 503 291-2569, Fax: 503 297-1090, http://www.pcisig.com/home
- *PCI Local Bus Specification, Revision 2.3*
  PCI Special Interest Group (PCI-SIG)
  5440 SW Westgate Drive #217, Portland, OR 97221 USA
  Tel: 503 291-2569, Fax: 503 297-1090, http://www.pcisig.com/home
- *PCI to PCI Bridge Architecture Specification*, *Revision 1.2*, June 9, 2003
  PCI Special Interest Group (PCI-SIG)
  5440 SW Westgate Drive #217, Portland, OR 97221 USA
  Tel: 503 291-2569, Fax: 503 297-1090, http://www.pcisig.com/home
- *PCI Bus Power Management Interface Specification, Revision 1.1*, December 18, 1998
  PCI Special Interest Group (PCI-SIG)
  5440 SW Westgate Drive #217, Portland, OR 97221 USA
  Tel: 503 291-2569, Fax: 503 297-1090, http://www.pcisig.com/home
- *PCI-X Addendum to PCI Local Bus Specification, Revision 1.0b*
  PCI Special Interest Group (PCI-SIG)
  5440 SW Westgate Drive #217, Portland, OR 97221 USA
  Tel: 503 291-2569, Fax: 503 297-1090, http://www.pcisig.com/home
- IEEE Standard 1149.1-1990, *IEEE Standard Test Access Port and Boundary-Scan Architecture*, 1990
  The Institute of Electrical and Electronics Engineers, Inc.
  445 Hoes Lane, PO Box 1331, Piscataway, NJ 08855-1331, USA
  Tel: 800 678-4333 (domestic only) or 732 981-0060, Fax: 732 981-1721, http://www.ieee.org/portal/index.jsp

Note: *In this data book, shortened titles are provided to the previously listed documents. The following table lists these abbreviations.*

Supplemental Documentation Abbreviations

Abbreviation	Document
PCI r2.1	*PCI Local Bus Specification, Revision 2.1*
PCI r2.3	*PCI Local Bus Specification, Revision 2.3*
P-to-P Bridge r1.2	*PCI to PCI Bridge Architecture Specification, Revision 1.2*
PCI Power Mgmt. r1.1	*PCI Bus Power Management Interface Specification, Revision 1.1*
PCI-X r1.0b	*PCI-X Addendum to PCI Local Bus Specification, Revision 1.0b*
IEEE Standard 1149.1-1990	*IEEE Standard Test Access Port and Boundary-Scan Architecture*

DATA ASSIGNMENT CONVENTIONS

Data Assignment Conventions

Data Width	PCI 6520 Convention
1 byte (8 bits)	Byte
2 bytes (16 bits)	Word
4 bytes (32 bits)	DWORD/Dword
8 bytes (64 bits)	QWORD/Qword

REVISION HISTORY

Date	Version	Comments
04/04	1.0	Production Release, Silicon Revision BB.
05/04	2.0	Production Release, Silicon Revision CB.

PCI 6520

Transparent PCI-X-to-PCI-X Bridge

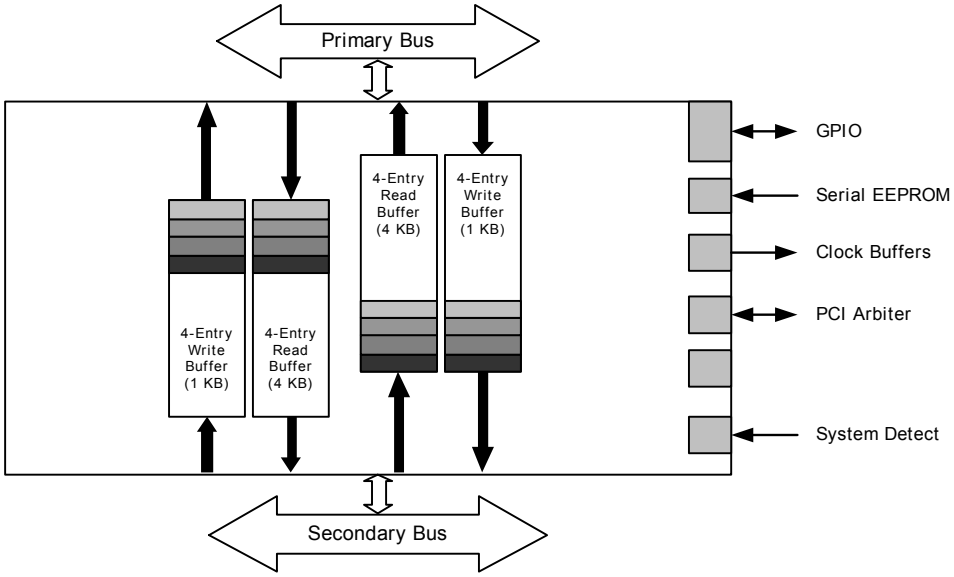
Asynchronous 64-Bit, 133 MHz PCI-X-to-PCI-X Bridge for Servers, Storage, Telecommunications, Networking, and Embedded Applications and High-Performance 10-KB Buffer PCI-X-to-PCI Bridging

May 2004
Version 2.0

FEATURE SUMMARY

The PLX FastLane™ PCI 6520 PCI-X-to-PCI-X Bridge is a device capable of 64-bit, 133 MHz operation. The device is designed for high-performance and high-availability uses such as PCI-X slot expansion, PCI-X-to-PCI conversion, multi-device attachment, and frequency conversion. The PCI 6520 has sophisticated buffer management and buffer configuration options designed to provide customizable performance for efficient throughput.

- *PCI-X r1.0b*-compliant at 64-bit, 133 MHz
  - Backward compatible with *PCI r2.3*
  - Support for input-pin-enforced PCI-X operation
- 5V tolerant I/O
- Asynchronous design for primary and secondary ports
  - 33 to 133 MHz operation
  - Either port may run at the higher frequency
- 8 GPIO pins with output control, with power-up status latch capabilities
- Primary port can be set to PCI-X protocol, without requiring the normal reset initialization sequence
- Flow-through, zero wait state bursts of up to 4 KB
  - Optimal for large volume Data transfer
- Supports up to four simultaneous Posted Writes and Delayed transactions in each direction
- Supports up to four simultaneous Split transactions in each direction in PCI-X mode
- Optional segmented 1-KB buffer for each of the four Read FIFO entries
- 10-KB buffers
  - 1-KB downstream Posted Write buffer
  - 1-KB upstream Posted Write buffer
  - 4-KB downstream Read Data buffer
  - 4-KB upstream Read Data buffer
- Configurable prefetch size of up to 2 KB
  - Ideal for PCI-to-PCI-X transfers
- 5 secondary clock outputs
  - Pin-controlled enable
  - Individual maskable control
- Supports downstream and upstream Lock
- Supports secondary port PCI/PCI-X Private Devices and Private Memory space (equivalent to Opaque memory)
- Reference clock input option
  - Primary and secondary port PCI-X frequency detection
- Serial EEPROM loadable
- Programmable *PCI Read-Only* register configurations
- Programmable arbitration for eight secondary bus masters
  - Optional External Arbiter
- PCI Mobile Design Guide and Power Management D_{3cold} Wakeup capable
- Enhanced address decoding
  - Supports 32-bit I/O address range
  - 32-bit Memory-Mapped I/O Address range
  - ISA-Aware mode for legacy support in the first 64 KB of I/O address range
  - VGA addressing and palette snooping support
- IEEE Standard 1149.1-1990 JTAG interface
- Low power 0.25µ CMOS process
- Industry standard 27 x 27 mm 380-pin (ball) PBGA package



PCI 6520 Block Diagram

1 INTRODUCTION

This section provides information about PLX Technology, Inc., and its products, the FastLane™ PCI 6000 Bridge Series, and PCI 6520 features and applications.

1.1 COMPANY AND PRODUCT INFORMATION

PLX Technology, Inc., is the leading supplier of standard interconnect silicon to the storage, communications, server, and embedded-control industries. PLX's comprehensive I/O interconnect product offering ranges from I/O accelerators, PCI-to-PCI bridges, PCI-X-to-PCI-X bridges, and HyperTransport™ bridges to the PLX PCI Express-based family of switches and bridges currently under development.

In addition to a broad product offering, PLX provides development tool support through Software Development Kits (SDKs), hardware Rapid Development Kits (RDKs), and third-party tool support through the PLX Partner Program. Our complete tool offering, combined with leadership PLX silicon, enables system designers to maximize system throughput, lower development costs, minimize system design risk, and provide faster time to market.

The PLX commitment to meeting customer requirements extends beyond complete product solutions, and includes active participation in industry associations. PLX contributes to the key standard-setting bodies in our industry, including PCI-SIG™ (the special interest group responsible for the creation and release of all PCI specifications), PICMG® (the organization responsible for CompactPCI and the new AdvancedTCA™ standard for fabrics), HyperTransport™ Consortium, and Blade Systems Alliance (BladeS). Furthermore, PLX is a key developer for PCI Express technology and a member of the Intel Developers Network for PCI Express Technology.

Founded in 1986, PLX has been developing products based on the PCI industry standard since 1994. PLX is publicly traded (NASDAQ:PLXT) and headquartered in Sunnyvale, CA, USA, with other domestic offices in Utah and Southern California. PLX European operations are based in the United Kingdom and Asian operations are based in China and Japan.

1.2 FASTLANE PCI 6000 BRIDGE SERIES

The PLX FastLane PCI 6000 series offers the industry's broadest set of PCI-to-PCI and PCI-X-to-PCI-X bridges. These bridges allow additional devices to be attached to the PCI Bus, and provide the ability to include intelligent adapters on a PCI Bus. In addition, these bridges allow PCI Buses of different speeds to be part of the same subsystem. (Refer to Table 1-1 and Figure 1-1.)

The PLX PCI and PCI-X family of interconnect products include both PCI-to-PCI and PCI-X-to-PCI-X bridging devices, offering system designers innovative features along with improved I/O performance. The PLX FastLane PCI 6000 series of PCI-to-PCI bridging products provide support for the entire range of current PCI Bus data widths and speeds, including 32-bit 33 MHz, 64-bit 66 MHz, and the latest 64-bit 133 MHz PCI-X variety of the standard.

The FastLane PCI 6000 product line is distinguished by featuring the widest range of options, lowest power requirements, highest performance, and smallest footprint in the industry. The product line includes features such as the ability to clock the PCI Bus segments asynchronously to one another.

The entire line of PLX bridging products are designed to provide high-performance interconnect for servers, storage, telecommunications, networking, and embedded applications. Like all PLX interconnect chips, the FastLane PCI 6000 series products are supported by PLX comprehensive reference design tools and the industry-recognized PLX support infrastructure.

Table 1-1. FastLane PCI 6000 Series PCI and PCI-X Bridge Product Comparison

Features	PCI 6140-AA33PC	PCI 6150-BB66BC PCI 6150-BB66PC	PCI 6152-CC33BC PCI 6152-CC33PC	PCI 6152-CC66BC	PCI 6156-DA33PC
PCI Bus Type	32-bit 33 MHz PCI	32-bit 66 MHz PCI	32-bit 33 MHz PCI	32-bit 66 MHz PCI	32-bit 33 MHz PCI
PCI Local Bus Support	r2.1 compliant	r2.3 compliant	r2.2 compliant	r2.2 compliant	r2.2 compliant
3.3 and 5V Tolerant I/O	Yes	Yes	Yes	Yes	Yes
Asynchronous Operation	No	25 to 66 MHz	No	No	No
Power Dissipation	200 mW	1.8W	300 mW	300 mW	300 mW
GPIO Interface	No	Four GPIO Pins	Four GPIO Pins	Four GPIO Pins	No
Transparency Modes	Transparent only	Transparent only	Transparent only	Transparent only	Transparent only
CompactPCI-Compatible Hot Swap	*Friendly*	r2.0 with PI = 1	*Friendly*	*Friendly*	—
Data FIFO	—	1 KB FIFO	—	—	—
Number of Bus Masters on Secondary Bus	Up to 4	Up to 9	Up to 4	Up to 4	Up to 10
Retry Architecture	Standard	Standard	Performance-Optimized	Performance-Optimized	Performance-Optimized
Programmable Flow-Through	—	Yes	—	—	—
Programmable Prefetch	Not specified	Up to 4 KB	N/A	N/A	N/A
Zero Wait State Burst	Up to 1 KB	Up to 1 KB	Up to 1 KB	Up to 1 KB	Up to 1 KB
Serial EEPROM Support	—	Yes	Yes	Yes	Yes
Vital Product Data Registers	—	Yes	Yes	Yes	Yes
D_3 Wakeup Power Management	Yes	Yes	Yes	Yes	Yes
Secondary Clock Outputs	Yes	Yes	Yes	Yes	Yes
JTAG Support	—	IEEE 1149.1 compliant	—	—	—
Packaging	PQFP-128	PBGA-256 PQFP-208	Tiny BGA-160 PQFP-160	Tiny BGA-160	PQFP-208
Package Size	23 x 17 mm	17 x 17 mm 31 x 31 mm	15 x 15 mm 32 x 32 mm	15 x 15 mm	31 x 31 mm
Rapid Development Kit	PCI 6140RDK	PCI 6150RDK	PCI 6152RDK	PCI 6152RDK	PCI 6156RDK

Table 1-1. FastLane PCI 6000 Series PCI and PCI-X Bridge Product Comparison (Continued)

Features	PCI 6350-AA66PC	PCI 6154-BB66BC	PCI 6254-BB66BC	PCI 6520-XX	PCI 6540-XX
PCI Bus Type	32-bit 66 MHz PCI	64-bit 66 MHz PCI	64-bit 66 MHz	64-bit 133 MHz PCI-X	64-bit 133 MHz PCI-X
PCI Local Bus Support	r2.2 compliant	r2.3 compliant	r2.3 compliant	r2.3 compliant	r2.3 compliant
3.3 and 5V Tolerant I/O	Yes	Yes	Yes	Yes	Yes
Asynchronous Operation	Yes	25 to 66 MHz	25 to 66 MHz	33 to 133 MHz	25 to 133 MHz
Power Dissipation	1.47W	2.0W	2.0W	1.0W	1.0W
GPIO Interface	Four GPIO Pins	Four GPIO Pins	16 GPIO Pins	8 GPIO Pins	16 GPIO Pins
Transparency Modes	Transparent only	Transparent only	Transparent, Non-Transparent and Universal modes	Transparent only	Transparent, Non-Transparent and Universal modes
CompactPCI-Compatible Hot Swap	—	—	r2.0 with PI = 1	—	r2.0 with PI = 1
Data FIFO	192 byte	1 KB FIFO	1 KB FIFO	10 KB FIFO	10 KB FIFO
Number of Bus Masters on Secondary Bus	Up to 9	Up to 9	Up to 9	Up to 8	Up to 8
Retry Architecture	Standard	Standard	Standard	Standard	Standard
Programmable Flow-Through	Yes	Yes	Yes	Yes	Yes
Programmable Prefetch	Up to 2 KB	Up to 4 KB	Up to 4 KB	Up to 4 KB	Up to 4 KB
Zero Wait State Burst	Up to 4 KB	Up to 1 KB	Up to 1 KB	Up to 4 KB	Up to 4 KB
Serial EEPROM Support	Yes	Yes	Yes	Yes	Yes
Vital Product Data Registers	Yes	Yes	Yes	Yes	Yes
D_3 Wakeup Power Management	Yes	Yes	Yes	Yes	Yes
Secondary Clock Outputs	Yes	Yes	Yes	Yes	Yes
JTAG Support	IEEE 1149.1 compliant	IEEE 1149.1 compliant	IEEE 1149.1 compliant	IEEE 1149.1 compliant	IEEE 1149.1 compliant
Packaging	PBGA-256 PQFP-208	PBGA-304	PBGA-365	PBGA-380	PBGA-380
Package Size	17 x 17 mm 31 x 31 mm	31 x 31 mm	31 x 31 mm	27 x 27 mm	27 x 27 mm
Rapid Development Kit	PCI 6350RDK	PCI 6154RDK	PCI 6254RDK	PCI 6520RDK	PCI 6540RDK

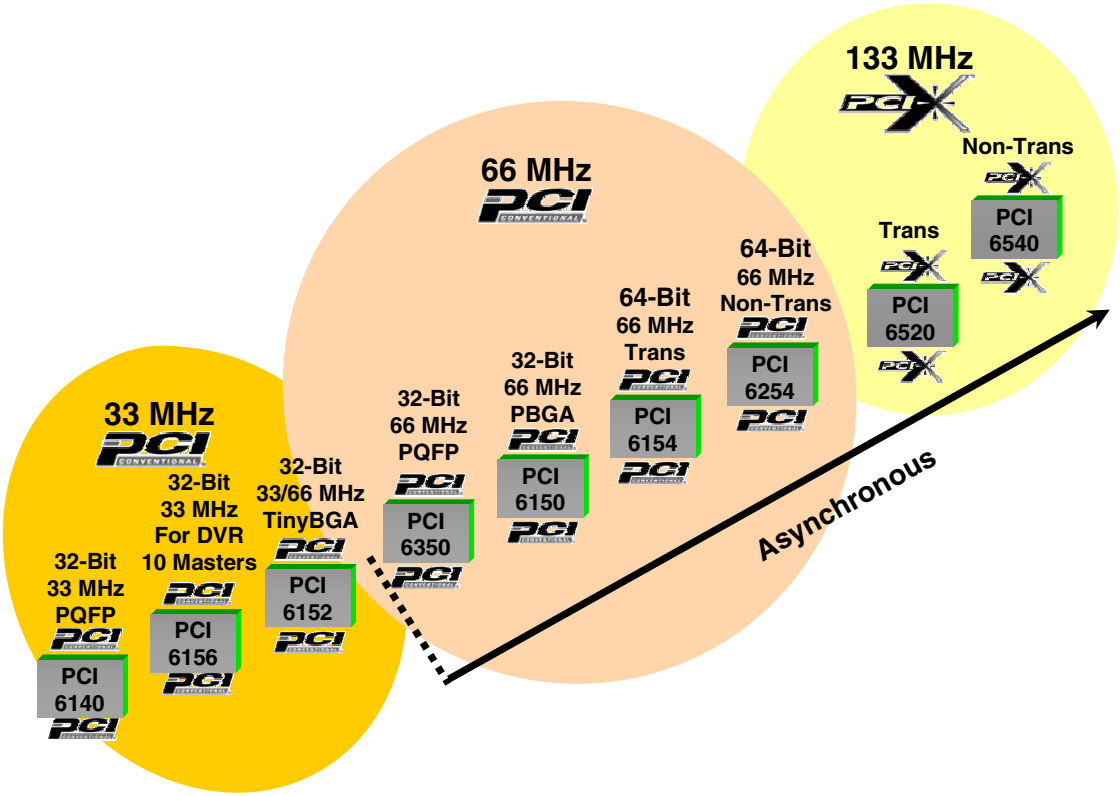


Figure 1-1. FastLane PCI 6000 Bridge Series

1.2.1 PCI 6520

As illustrated in Figure 1-2, the PCI 6520 is a two-port device providing full *asynchronous* operation between the *primary* and *secondary* ports. The secondary bus may be run faster than the primary bus, and vice versa.

PCI-X is the next generation of the industry-standard PCI Bus. This standard provides complete backward compatibility in terms of electrical characteristics, software, and form factor, and when used in PCI-X mode, the maximum transfer speed increases to 133 MHz, allowing a maximum bursting bandwidth of 1 GBps. Alternatively, PCI-X allows greater electrical expansion of the PCI Bus at 66 MHz by registering Data transfers.

A *transparent* PCI bridge is meant to provide electrical isolation to the system. It allows additional loads (and devices) to be attached to the bus, and can also be used to operate dissimilar PCI Bus data widths and speeds on the same system. *For example*, a transparent bridge can allow several 64-bit, 66 MHz PCI devices to attach to a 133 MHz PCI-X slot.

An *asynchronous* bridge provides the ability to run each port from a completely independent clock. This allows the system designer to provide the highest performance on each side of the bridge, without forcing one side to slow down based on a slower device on the other side of the bridge. The advantage of an asynchronous bridge is that the two clock domains can be arbitrarily different, and not based on a synchronous version of the other clock.

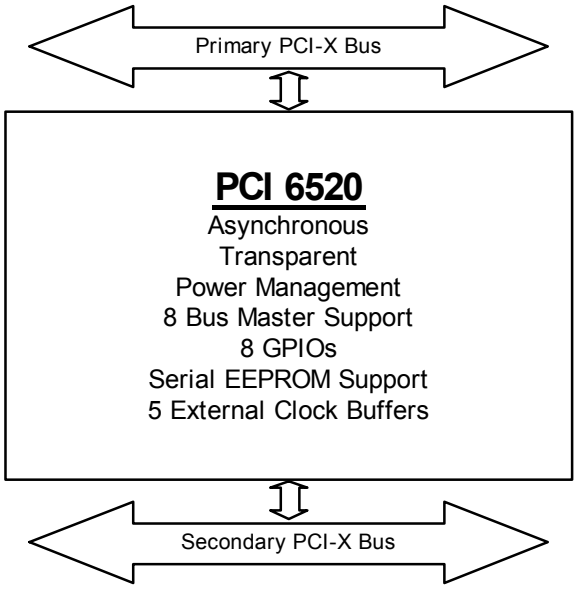


Figure 1-2. PCI 6520 PCI-X-to-PCI-X Bridge

1.3 FEATURE DESCRIPTION

The PCI 6520 provides a range of added-value features to system designers, including:

- Two PCI-X ports, each capable of running at the full 64-bit, 133 MHz speed
  - Backward compatible to *PCI r2.3*
- Asynchronous primary and secondary ports
  - Ports can operate from 33 MHz to 133 MHz
  - Either port can operate slower or faster than the other port
- 5V tolerant I/O
- Programmable prefetch
- Programmable Flow Through
- Zero wait state burst
- 10-KB Data FIFO
- 5 secondary clock outputs
- Reference clock input for frequency detection
- PCI Power Management support
- Arbitration support for eight secondary bus masters
- Serial EEPROM for configuration
- 16 General Purpose I/O pins
- Vital Product Data (VPD)
- JTAG boundary scan

1.4 APPLICATIONS

1.4.1 Multiple Device Expansion

Figure 1-3 illustrates the PCI 6520 being used to provide electrical isolation to the PCI-X Bus. This is necessary because PCI-X slots restrict the number of loads that can be accommodated. The devices on the secondary port can be *PCI* or *PCI-X*, and the bus can operate at a range of frequencies and bus data widths. This configuration is a common mechanism for providing multiple, high-speed Ethernet or Fibre Channel connections on a single PCI-X card.

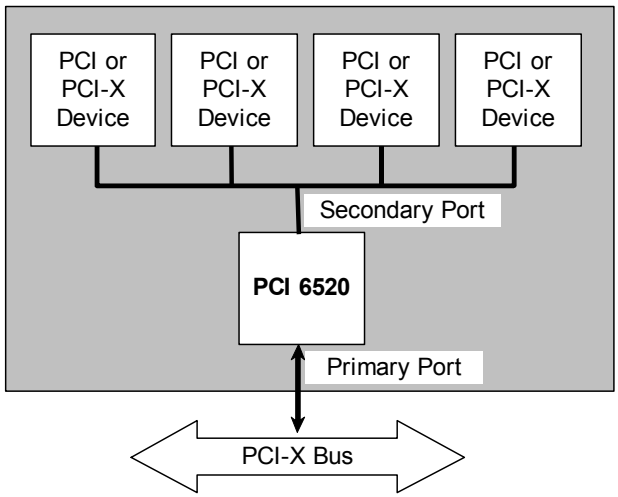


Figure 1-3. Multiple Device Expansion

2 FUNCTIONAL OVERVIEW

This section describes general operation of the PCI 6520 bridge, and provides an overview of write and read transactions.

2.1 GENERAL OPERATION

As illustrated in Figure 2-1, the PCI 6520 uses programmable buffers to regulate data flow between the primary and secondary ports. There are two sets of buffers—one for *downstream* commands (data flows from primary-to-secondary bus) and one for *upstream* commands (data flows from secondary-to-primary bus). The buffers are organized as follows:

- 4-Entry Write buffer (1 KB)
- 4-Entry Read buffer (4 KB)

Each PCI and/or PCI-X port can run at different (asynchronous) frequencies, which allows the designer to optimize the performance of each bus. Both the primary and secondary PCI and/or PCI-X ports may be operated at 32- or 64-bit bus data widths, and the two buses may be of different widths.

The PCI 6520 provides an Internal Arbiter function on the secondary bus, for up to eight secondary bus masters. However, the Internal Arbiter may be disabled if an External Arbiter is used. The PCI 6520 also sources five secondary PCI and/or PCI-X clock outputs.

The PCI 6520 provides features satisfying the requirements of *PCI Power Mgmt. r1.1*, supporting Power Management states D_0 through D_{3cold} and D_{3hot}. (Refer to Section 20, “Power Management,” for further details.)

The PCI 6520 supports a serial EEPROM device for register configuration data. This allows the PCI 6520 to automatically load custom configuration upon power-up, which minimizes the software overhead of configuring the bridge through a host processor.

The PCI 6520 fully supports Vital Product Data (VPD) by providing the Address, Data, and Control registers (PVPDAD; PCI:EAh, PVPDATA; PCI:ECh, PVPDID; PCI:E8h, and PVPD_NEXT; PCI:E9h) for accessing VPD stored in the unused portion of the serial EEPROM. VPD allows reading or writing of user data to the upper 192 bytes of serial EEPROM space, and that data can contain information such as board serial number, software revision, firmware revision, or other data required for non-volatile storage. (Refer to Section 21, “VPD,” for further details.)

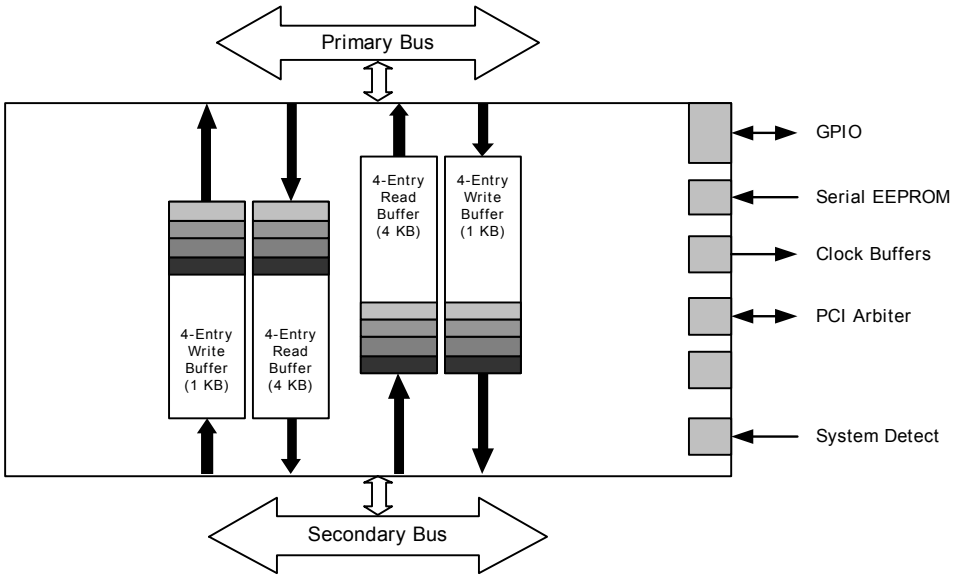


Figure 2-1. PCI 6520 Block Diagram

2.2 WRITE TRANSACTIONS

The primary or secondary bus accomplishes a Write operation by placing the address and data into the *Write buffer*. This initiates a PCI Write operation on the other bus. The Write operation is called a *Posted Write* operation, because the initiating bus performs the write, then moves on without waiting for the operation to complete.

The PCI 6520 provides the ability to combine Write operations when the operations are directed at the same Address range. The device recognizes when Write operations are directed at consecutive addresses, and accumulates and bursts those Write operations to the PCI Bus for increased bandwidth.

In addition, the PCI 6520 has the capability to start a Write operation before receiving all Write data. In this case, the Write operation begins when there is sufficient Write data to begin the burst, providing a Flow-Through operation as the balance of the Write data arrives in the device.

2.3 READ TRANSACTIONS

When the downstream or upstream bus needs to read data from the other bus, the bus places the Read request into the *Read Command queue*. This initiates a Read operation on the other bus, and the data is placed into the associated *Read buffer* as it returns. For PCI-X transactions, the PCI 6520 also prefetches data as defined in the original Read operation.

For PCI transactions, there is an additional prefetch mechanism when returning the requested Read data. In this mode, the PCI 6520 can be programmed to prefetch up to 2 KB of data at a time. This data is stored in the Read buffer and is not flushed until the buffer times out. If requested, prefetched data can be delivered to the PCI Bus without the normal read on the other bus.

3 PIN DESCRIPTION

This section describes the PCI 6520 pins (balls), including pin summary, pull-up and pull-down resistor recommendations, power supply de-coupling, and pinout listings.

3.1 PIN SUMMARY

Tables 3-4 through Table 3-12 describe each PCI 6520 pin:

- PCI-X Primary Bus Interface
- PCI-X Secondary Bus Interface
- Clock-Related
- Reset
- JTAG
- Serial EEPROM Interface
- GPIO
- Miscellaneous
- Power, Ground, and No Connect

For a visual view of the PCI 6520 pinout, refer to Section 24, “Mechanical Specs.”

Table 3-1 lists abbreviations used in Section 3 to represent various pin types.

Table 3-1. Pin Type Abbreviations

Abbreviation	Pin Type
I	CMOS Input (5V input tolerant, I/O V_{DD}=3.3V)
I/O	CMOS Bi-Directional Input Output (5V input tolerant, I/O V_{DD}=3.3V)
O	CMOS Output
OD	Open Drain
OZ	Output Three-State
PCI	PCI/PCI-X Compliant
PI	PCI-X Input (5V input tolerant, I/O V_{DD}=3.3V)
PO	PCI-X Output
PU	Signal is internally pulled up
PSTS	PCI-X Sustained Three-State Output, Drive High for One CLK before Floating (5V input tolerant, I/O V_{DD}=3.3V)
PTS	PCI-X Three-State Bi-Directional (5V input tolerant, I/O V_{DD}=3.3V)

3.2 PULL-UP AND PULL-DOWN RESISTOR RECOMMENDATIONS

Pull-up and pull-down resistor values are not critical. With the exception of those mentioned in Section 3.2.1, a 10K-Ohm resistor is recommended unless stated otherwise.

3.2.1 PCI Bus Interface Pins

The pins detailed in Table 3-2 are generic primary and secondary PCI interface pins. When producing motherboards, system slot cards, adapter cards, backplanes, and so forth, the termination of these pins should follow the guidelines detailed in *PCI r2.3* and *PCI-X r1.0b*.

Table 3-2. Generic PCI Bus Interface Pins that follow *PCI r2.3* and *PCI-X r1.0b* Layout Guidelines

Bus	Pin Name
Primary	P_ACK64#, P_AD[63:0], P_CBE[7:0]#, P_DEVSEL#, P_FRAME#, P_GNT#, P_IDSEL, P_IRDY#. P_LOCK#, P_M66EN, P_PAR, P_PAR64, P_PERR#, P_REQ#, P_REQ64#, P_SERR#, P_STOP#, P_TRDY#
Secondary	S_ACK64#, S_AD[63:0], S_CBE[7:0]#, S_DEVSEL#, S_FRAME#, S_GNT[7:0]#, S_IRDY#. S_LOCK#, S_M66EN, S_PAR, S_PAR64, S_PERR#, S_REQ[7:0]#, S_REQ64#, S_SERR#, S_STOP#, S_TRDY#

The following guidelines are not exhaustive and should be read in conjunction with the appropriate sections of *PCI r2.3* and *PCI-X r1.0b*.

PCI control signals require a pull-up resistor on the motherboard to ensure that these signals are always at valid values when a PCI Bus agent is not driving the bus. These control signals include ACK64#, DEVSEL#, FRAME#, IRDY#, LOCK#, PERR#, REQ64#, SERR#, STOP#, and TRDY#. The 32-bit point-to-point and shared bus signals do ***not*** require pull-up resistors, as bus parking ensures that these signals remain stable. The other 64-bit signals—AD[63:32], CBE[7:4]# and PAR64—also require pull-up resistors, as these signals are not driven during 32-bit transactions. Depending on the application, M66EN may also require a pull-up resistor. The value of these pull-up resistors depends on the bus loading. *PCI r2.3* provides formulas for calculating these resistors.

When making adapter cards in which the PCI 6520 primary port is wired to the PCI connector, pull-up resistors are not required because they are pre-installed on the motherboard.

Based on the above, in an embedded design, pull-up resistors may be required for PCI control signals on the primary and secondary buses. Whereas, for a PCI adapter card design, pull-up resistors are required only on the PCI 6520 port that is not connected to the motherboard or host system.

S_REQ[7:1]# inputs must be pulled high with a 10K-Ohm pull-up resistor. S_REQ0# also requires a 10K-Ohm pull-up resistor if S_CFN#=0.

3.2.2 Clock-Related Pins

Clock routing is detailed in Section 4, "Clocking." Pull-up resistors are not required on the S_CLKO[4:0] pins; however, a series termination resistor is required when using these pins. S_CLKO0 may require a pull-up resistor when this pin is used as a result of S_CLKOFF=1, or disabled (CLKCNTRL [1:0]=11b; PCI:68h). S_CLKO[4:0] may also require pull-up resistors if they are disabled by pulling MSK_IN high. Table 3-3 delineates the remaining clock pin resistor requirements.

Table 3-3. Clock Pin Pull-Up/Pull-Down Resistor Requirements

Resistor Requirements	Pin Name
Must pull low	P_CLKOE, P_CR, S_CR, P_PLLEN#*
Pull or tie low or high, or connect to 3.3V power supply	S_CLKIN_STB, S_PLLEN#*
Pull high or low if unused	OSCIN, REFCLK
Optionally pull high or low	MSK_IN, OSCSEL#, S_CLKOFF
Pull-up or pull-down resistor not required	P_CLKIN, S_CLKIN, S_CLKO[4:1]**

Notes: ** Refer to Section 4.6, "PLL and Clock Jitter," for further details regarding P_PLLEN# and S_PLLEN# when used in low frequency applications.*

*** Refer also to the text preceding this table.*

3.2.3 Reset Pins

The P_RSTIN# Reset signal may require a pull-up resistor, depending on the application. The S_RSTOUT# Reset signal does ***not*** require a pull-up nor pull-down resistor.

The PWRGD signal should be pulled to 3.3V or driven high when the 3.3V supply is stable. It should not be connected to 5V.

Note: *PWRGD requires a clean, low-to-high transition. The PWRGD input is not internally de-bounced; therefore, this input must be pulled up to 3.3V, rather than 5V.*

3.2.4 JTAG/Boundary Scan Pins

The TCK, TDI, and TMS JTAG signals must be externally pulled high or low to a known state. The TDO signal must be externally pulled high. The TRST# signal must be pulled low, using a 330-Ohm resistor.

3.2.5 Serial EEPROM Pins

EEPCLK does ***not*** require a pull-up nor pull-down resistor. EEPDATA requires an external pull-up resistor.

3.2.6 GPIO Pins

When programmed as outputs, the GPIO[7:0] pins do not require external pull-up nor pull-down resistors. If configured as inputs, pull the GPIO[7:0] pins high or low, depending on the application.

3.2.7 Miscellaneous Pins

The BPCC_EN, DEV64#, and PRV_DEV signals may optionally be pulled high or low. S_CFN# may also optionally be pulled high or low, but must be tied low to use the Internal Arbiter.

P_XCAP is normally configured by the host. In an embedded system, the P_XCAP pin may be pulled high or low. When P_XCAP is pulled high, the primary port is forced to use PCI-X Bus protocol. In general, pull the P_XCAP pin low. When P_XCAP is pulled low, the primary port bus protocol is set by the PCI-X Initialization Pattern when P_RSTIN# is de-asserted.

Tie S_XCAP_PU to S_XCAP_IN by way of a 1K-Ohm resistor. S_XCAP_IN should follow the Programmable Pull-Up and Binary Input Buffer method, as defined in *PCI-X r1.0b,* "Detection of PCI-X Add-in Card Capability" appendix. *PCI-X r1.0b* allows detection of Conventional PCI, PCI-X 66 MHz, PCI-X 100 MHz (with PCIX100MHZ=1), or PCI-X 133 MHz (with PCIX100MHZ=0) adapter cards or devices on the secondary interface.

P_TST1 and S_TST1 should be pulled high and P_TST0 and S_TST0 should be pulled low.

3.2.8 System Voltage Pins

For designs and add-in cards that have an independent V_{IO} voltage source, and for which proper power sequencing cannot be guaranteed, the current between the V_{IO} voltage source and PCI 6520 V_{IO} pins must be limited to protect the device from long-term undue stress resulting in damage (*such as* from resistor insertion).

Note: *By their nature, add-in cards cannot assume proper power sequencing and requirements must be met by system power supplies.*

Use the following guidelines to determine the required resistance value for the P_V_{IO} and S_V_{IO} pins:

- **3.3V signaling environments**—40 to 200-Ohm resistance between the V_{IO} voltage source and the PCI 6520 V_{IO} pins is recommended if V_{IO} is a maximum of 3.6V
- **3.3 or 5V signaling environments**—40 to 70-Ohm resistance is recommended

A single resistor can be used if the V_{IO} pins are bused, or multiple parallel resistors can be used between the V_{IO} voltage source and V_{IO} pins. The resistor power dissipation rating depends upon the resistance size and signaling environment. *For example,* if a single 50-Ohm resistor is used in a 5V signaling environment, the worst-case power dissipation would result in 480 mW. (Refer to Figure 3-1.)

If four, 200-Ohm resistors are used in parallel, each would be required to dissipate 120 mW.

Any resistance value within the recommended ranges prevents the device from being damaged, while providing sufficient clamping action to keep the Input Voltage (V_{IN}) below its maximum rating. A resistance value at the lower end of the range is recommended to provide preferable clamping action, and a sufficient V_{IN} margin.

$$480\ \text{mW} = \frac{(V \bullet V) / R\ (5.5V\ (\text{maximum signal amplitude, plus } 10\%) - 0.6V\ (1\ \text{diode drop}))^2}{50\ \text{Ohms}}$$

Figure 3-1. Worst-Case Power Dissipation Example

3.3 POWER SUPPLY DE-COUPLING

De-couple all V_{DD_CORE}, V_{DD_IO}, P_AV_{DD}, P_V_{IO}, S_AV_{DD}, and S_V_{IO} lines. The de-coupling level depends on power plane routing and the acceptable power supply noise level. In an ideal case, de-couple all previously listed power supply pins, using a parallel combination of 10 and 100 nF capacitors. Use of the 10 nF capacitors is due to the relatively high inductance of 100 nF capacitors, which can prevent the capture of fast transients.

Due to routing constraints, it may not be possible to add the parallel combination to all supply pins. In this case, the 10 and 100 nF capacitors can be used alternately among the supply pins.

Low-inductance 100 nF capacitors are available, which may be used in place of the 10 nF/100 nF parallel combination.

Take care when choosing the capacitor material. Some types have poor thermal characteristics, resulting in substantial drops in the capacitance value at higher temperatures.

Connect de-coupling capacitors to the appropriate ground plane. Do ***not*** de-couple digital supplies to the clean analog Phase-Locked Loop (PLL) grounds or vice versa.

Phase-locked loops are sensitive to power and ground noise. Using the same supply/ground for more than one PLL, or the digital supply/ground (Core or I/O Ring) for either PLL is not recommended, as noise coupling into the PLL supply/ground can cause PLL malfunction.

Each PLL requires a dedicated and independent power supply and ground. Ideally, de-couple each PLL supply with 100 pF, 47 nF, and 10 µF capacitors, in parallel. One set of capacitors is required for each PLL supply.

In addition to the above, a 10 µF bulk de-coupling capacitor for the digital supply is also recommended. The number and placement of this capacitor depends on the power supply and board design.

3.4 PINOUT

Note: *Refer to Section 3.2 for pull-up and pull-down resistor recommendations not specifically stated in these tables.*

Table 3-4. Primary PCI/PCI-X Bus Interface Pins

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
P_ACK64#	Primary 64-Bit Transfer Acknowledge	1	I/O PSTS PCI	W14	When asserted by the target device, indicates that the target can perform 64-bit Data transfers. Uses the same timing as P_DEVSEL#. When de-asserting, driven high for one cycle before being placed into a high-impedance state.
P_AD[31:0]	Primary Address and Data, Lower 32 Bits	32	I/O PTS PCI	R4, R2, R1, T4, T3, T2, T1, U4, U1, V2, V1, W2, W1, V7, W7, Y7, W10, Y10, T11, U11, V11, W11, Y11, T12, W12, Y12, U13, V13, W13, Y13, U14, V14	Multiplexed Address and Data Bus. Provides the lower 32 Address and/or Data pins. Address is indicated by P_FRAME# assertion during PCI and/or PCI-X transactions. Write data is stable and valid when P_IRDY# is asserted and Read data is stable and valid when P_TRDY# is asserted. Data is transferred on rising clock edges when P_IRDY# and P_TRDY# are asserted. During bus idle, driven by the PCI 6520 to valid logic levels when P_GNT# is asserted. Additionally, these lines provide a portion of the attribute during the Attribute phase of PCI-X transactions. (Refer to Section 14, “PCI Bus Arbitration,” for further details.)
P_AD[63:32]	Primary Address and Data, Upper 32 Bits	32	I/O PTS PCI	V16, W16, Y16, U17, V17, W17, Y17, Y18, W18, U18, Y19, W19, V19, W20, V20, U19, U20, T17, T18, T19, T20, R17, R19, R20, P17, P18, P19, P20, N17, N18, N19, N20	Multiplexed Address and Data Bus. Provides the upper 32 Address and/or Data pins. During an Address phase (when using the DAC command and P_REQ64# is asserted), the upper 32 bits of a 64-bit address are transferred; otherwise, these bits are undefined. During a Data phase, the upper 32 bits of data are transferred if a 64-bit transaction is negotiated by P_REQ64# and P_ACK64# assertion.
P_CBE[3:0]#	Primary Lower Command and Byte Enables	4	I/O PTS PCI	U3, U8, V10, U12	Multiplexed Command and Byte Enable fields. Provides the transaction type during the PCI and/or PCI-X Address phase. In the Data phase of PCI and/or PCI-X Memory Write transactions, P_CBE[3:0]# provide Byte Enables. During the PCI-X Attribute phase, these signals provide a portion of the attribute information. During bus idle, the PCI 6520 drives P_CBE[3:0]# to valid logic levels when P_GNT# is asserted.

Table 3-4. Primary PCI/PCI-X Bus Interface Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
P_CBE[7:4]#	Primary Upper Command and Byte Enables	4	I/O PTS PCI	U15, W15, Y15, U16	Multiplexed Command and Byte Enable fields. During an Address phase (when using the DAC command and P_REQ64# is asserted), the bus command is transferred on these pins; otherwise, P_CBE[7:4]# are ***reserved*** and indeterminate. If a 64-bit transaction is negotiated by P_REQ64# and P_ACK64# assertion, then during a PCI and/or PCI-X Memory Write transaction Data phase, these pins indicate which byte lanes carry meaningful data.
P_DEVSEL#	Primary Device Select	1	I/O PSTS PCI	T9	Asserted by the target, indicating that the device is accepting the transaction. As a master, the PCI 6520 waits for P_DEVSEL# assertion within five cycles of P_FRAME# assertion; otherwise, the transaction terminates with a Master Abort. Before being placed into a high-impedance state, P_DEVSEL# is driven to a high state for one cycle.
P_FRAME#	Primary Frame	1	I/O PSTS PCI	V8	Driven by the initiator of a transaction to indicate the beginning and duration of an access. P_FRAME# de-assertion indicates the final Data phase requested by the initiator. Before being placed into a high-impedance state, P_FRAME# is driven to a high state for one cycle.
P_GNT#	Primary Grant	1	PI	P4	When asserted, the PCI 6520 can access the primary bus. During bus idle with P_GNT# asserted, the PCI 6520 drives P_AD*x*, P_CBE*x*#, P_PAR, and P_PAR64 to valid logic levels.
P_IDSEL	Primary Initialization Device Select	1	PI	U2	Used as a Chip Select line for Configuration accesses to PCI 6520 Configuration space and Configuration spaces behind the PCI 6520 bridge.
P_IRDY#	Primary Initiator Ready	1	I/O PSTS PCI	W8	Driven by the initiator of a transaction to indicate its ability to complete the current Data phase on the primary bus. Before being placed into a high-impedance state, P_IRDY# is driven to a de-asserted state for one cycle.
P_LOCK#	Primary Lock	1	I/O PSTS PCI	W9	Asserted by the bus master, indicating an atomic operation that may require multiple transactions to complete.

Table 3-4. Primary PCI/PCI-X Bus Interface Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
P_M66EN	Primary 66 MHz Enable	1	PI	M20	Set high to allow 66 MHz primary bus operation. Along with S_M66EN, controls the frequency output to the S_CLKO[4:0] pins. (Refer to Section 4, "Clocking," for further details.) Also used with the PCI-X initialization pattern to automatically enable and disable the primary PLL.
P_PAR	Primary Parity, Lower 32 Bits	1	I/O PTS PCI	U10	Parity is even across P_AD[31:0], P_CBE[3:0]#, and P_PAR [*such as*, an even number of ones (1)]. P_PAR is an input and valid and stable for one cycle after the Address phase (indicated by P_FRAME# assertion) for address parity. During Write transactions, P_PAR is an output and valid for one clock after P_TRDY# assertion. P_PAR is placed into a high-impedance state one cycle after the P_AD[31:0] lines are placed into a high-impedance state. During bus idle, the PCI 6520 drives P_PAR to a valid logic level when P_GNT# is asserted. During Read transactions, P_PAR is an input and valid for one clock after P_IRDY# assertion.
P_PAR64	Primary Parity, Upper 32 Bits	1	I/O PTS PCI	M16	Parity is even across P_AD[63:32] and P_CBE[7:4]#. P_PAR64 must be valid for one clock after each Address phase on transactions in which P_REQ64# is asserted. For Data phases, after P_PAR64 is valid, P_PAR64 remains valid until one Clock cycle after the current Data phase completes.
P_PERR#	Primary Parity Error	1	I/O PSTS PCI	Y9	Asserted when a Data Parity error is detected for data received on the primary interface. Before being placed into a high-impedance state, P_PERR# is driven to a de-asserted state for one cycle.
P_REQ#	Primary Request	1	OZ	P1	Asserted by the PCI 6520 to request ownership of the primary bus to perform a transaction. The PCI 6520 de-asserts P_REQ# for at least two PCI Clock cycles before re-asserting it. (Refer to Section 14.2, "Primary PCI Bus Arbitration," for further details.)

Table 3-4. Primary PCI/PCI-X Bus Interface Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
P_REQ64#	Primary 64-Bit Transfer Request	1	I/O PSTS PCI	Y14	***Must*** be asserted low by the *central resource* (bus support functions supplied by the host system) during reset to indicate the primary bus is a 64-bit bus. Asserted with P_FRAME# by a PCI Bus master to request a 64-bit Data transfer. The PCI 6520 ignores this input during reset. The PCI 6520 asserts P_REQ64# when the secondary PCI master performs a 64-bit transfer or the FORCE64 option is enabled (MSCOPT[15 and/or 11]=1; PCI:46h). ***Notes:*** *If P_REQ64# is high during reset, the primary bus is a 32-bit bus, and the PCI 6520* ***must*** *park the P_AD[63:32], P_CBE[7:4]#, and P_PAR64 signals.* *If there is no central resource in the bus to assert P_REQ64# low during reset, the PCI 6520 performs a 32-bit only data transfer and ignores P_REQ64# on the bus. A three-state buffer with input pin to ground, three-state select pin to bus reset and output pin to P_REQ64# can assert P_REQ64# during reset.*
P_SERR#	Primary System Error	1	OD	T10	P_SERR# can be driven low by any device to indicate a System error condition. The PCI 6520 drives P_SERR# if one of the following conditions is met: • Address Parity error • Posted Write Data Parity error on target bus • S_SERR# is asserted • Master Abort during Posted Write transaction • Target Abort during Posted Write transaction • Posted Write transaction discarded • Delayed Write request discarded • Delayed Read request discarded • Delayed transaction Master Timeout
P_STOP#	Primary Stop	1	I/O PSTS PCI	U9	Asserted by the target to end the transaction on the current Data phase. Before being placed into a high-impedance state, P_STOP# is driven to a de-asserted state for one cycle.

Table 3-4. Primary PCI/PCI-X Bus Interface Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
P_TRDY#	Primary Target Ready	1	I/O PSTS PCI	Y8	Driven by the target of a transaction to indicate its ability to complete the current Data phase on the primary bus. Before being placed into a high-impedance state, P_TRDY# is driven to a de-asserted state for one cycle.
Total		88			

Table 3-5. Secondary PCI/PCI-X Bus Interface Pins

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
S_ACK64#	Secondary 64-Bit Transfer Acknowledge	1	I/O PSTS PCI	B17	When asserted by the target device, indicates that the target can perform 64-bit Data transfers. Uses the same timing as S_DEVSEL#. When de-asserting, driven high for one cycle before being placed into a high-impedance state.
S_AD[31:0]	Secondary Address and Data, Lower 32 Bits	32	I/O PTS PCI	A6, B6, C6, D6, A7, B7, C7, D7, C8, D8, A9, B9, D9, E9, A10, B10, E12, A13, B13, C13, D13, A14, B14, C14, A15, B15, D15, A16, B16, C16, D16, A17	Multiplexed Address and Data Bus. Provides the lower 32 Address and/or Data pins. Address is indicated by S_FRAME# assertion during PCI and/or PCI-X transactions. Write data is stable and valid when S_IRDY# is asserted and Read data is stable and valid when S_TRDY# is asserted. Data is transferred on rising clock edges when S_IRDY# and S_TRDY# are asserted. Driven to a low logic level (0) when S_RSTOUT# is asserted. During bus idle, driven by the PCI 6520 to valid logic levels when the PCI 6520 is granted secondary bus ownership. Additionally, these lines provide a portion of the attribute during the Attribute phase of PCI-X transactions. (Refer to Section 14, “PCI Bus Arbitration,” for further details.)
S_AD[63:32]	Secondary Address and Data, Upper 32 Bits	32	I/O PTS PCI	B20, C20, C19, D20, D19, D18, D17, E20, E19, E18, E17, F20, F19, F17, G20, G19, G18, G17, H20, H19, H18, H17, J20, J19, J17, J16, K20, K19, K18, K17, K16, L20	Multiplexed Address and Data Bus. Provides the upper 32 Address and/or Data pins. During an Address phase (when using the DAC command and S_REQ64# is asserted), the upper 32 bits of a 64-bit address are transferred; otherwise, these bits are undefined. During a Data phase, the upper 32 bits of data are transferred if a 64-bit transaction is negotiated by S_REQ64# and S_ACK64# assertion. Driven to a low logic level (0) when S_RSTOUT# is asserted.

Table 3-5. Secondary PCI/PCI-X Bus Interface Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
S_CBE[3:0]#	Secondary Lower Command and Byte Enables	4	I/O PTS PCI	B8, C10, D12, D14	Multiplexed Command and Byte Enable fields. Provides the transaction type during the PCI and/or PCI-X Address phase. In the Data phase of PCI and/or PCI-X Memory Write transactions, S_CBE[3:0]# provide the Byte Enables. During the PCI-X Attribute phase, these signals provide a portion of the attribute information. Driven to a low logic level (0) when S_RSTOUT# is asserted. During bus idle, the PCI 6520 drives S_CBE[3:0]# to valid logic levels when the PCI 6520 is granted secondary bus ownership. (Refer to Section 14, "PCI Bus Arbitration," for further details.)
S_CBE[7:4]#	Secondary Upper Command and Byte Enables	4	I/O PTS PCI	A18, B18, A19, B19	Multiplexed Command and Byte Enable fields. During an Address phase (when using the DAC command and S_REQ64# is asserted), the bus command is transferred on these pins; otherwise, S_CBE[7:4]# are ***reserved*** and indeterminate. If a 64-bit transaction is negotiated by S_REQ64# and S_ACK64# assertion, then during a PCI and/or PCI-X Memory Write transaction Data phase, these pins indicate which byte lanes carry meaningful data. Driven to a low logic level (0) when S_RSTOUT# is asserted.
S_DEVSEL#	Secondary Device Select	1	I/O PSTS PCI	B11	Asserted by the target, indicating that the device is accepting the transaction. As a master, the PCI 6520 waits for the S_DEVSEL# assertion within five cycles of S_FRAME# assertion; otherwise, the transaction terminates with a Master Abort. Before being placed into a high-impedance state, S_DEVSEL# is driven to a high state for one cycle.
S_FRAME#	Secondary Frame	1	I/O PSTS PCI	D10	Driven by the initiator of a transaction to indicate the beginning and duration of an access. S_FRAME# de-assertion indicates the final Data phase requested by the initiator. Before being placed into a high-impedance state, S_FRAME# is driven to a high state for one cycle.
S_GNT0#	Secondary Grant 0	1	If S_CFN#=0 OZ otherwise PI	E1	Asserted by the PCI 6520 to grant the secondary bus to a secondary bus master when using internal arbitration (S_CFN#=0). When external arbitration is active (S_CFN#=1), S_GNT0# becomes the external bus Grant input to the PCI 6520. During bus idle with S_GNT0# asserted, the PCI 6520 drives S_AD*x*, S_CBE*x*#, S_PAR, and S_PAR64 to valid logic levels.

Table 3-5. Secondary PCI/PCI-X Bus Interface Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
S_GNT[7:1]#	Secondary Internal Arbiter Grant 7 to 1	7	If S_CFN#=0 OZ otherwise PI	G2, G3, G4, F1, F2, F3, F4	Asserted by the PCI 6520 to grant the secondary bus to a secondary bus master. The PCI 6520 de-asserts S_GNT[7:1]# for at least two PCI Clock cycles before re-asserting them. Pull S_GNT[7:1]# high if S_CFN#=1.
S_IRDY#	Secondary Initiator Ready	1	I/O PSTS PCI	E10	Driven by the initiator of a transaction to indicate its ability to complete the current Data phase on the secondary bus. Before being placed into a high-impedance state, S_IRDY# is driven to a de-asserted state for one cycle.
S_LOCK#	Secondary Lock	1	I/O PSTS PCI	D11	Asserted by the bus master, indicating an atomic operation that may require multiple transactions to complete.
S_M66EN	Secondary 66 MHz Enable	1	If P_M66EN=0 OD otherwise PI	L16	Driven low if P_M66EN is low; otherwise, driven from outside to select 66 or 33 MHz. S_M66EN ***must*** be pulled high with a 10K-Ohm resistor. Along with P_M66EN, controls the frequency output to the S_CLKO[4:0] pins. (Refer to Section 4, “Clocking,” for further details.)
S_PAR	Secondary Parity, Lower 32 Bits	1	I/O PTS PCI	B12	Parity is even across S_AD[31:0], S_CBE[3:0]#, and S_PAR [*such as*, an even number of ones (1)]. S_PAR is an input and valid and stable one cycle after the Address phase (indicated by S_FRAME# assertion) for address parity. During Write transactions, S_PAR is an output and valid for one clock after S_TRDY# assertion. S_PAR is placed into a high-impedance state one cycle after the S_AD[31:0] lines are placed into a high-impedance state. Driven to a low logic level (0) when S_RSTOUT# is asserted. During bus idle, the PCI 6520 drives S_PAR to a valid logic level when the PCI 6520 is granted secondary bus ownership. (Refer to Section 14, “PCI Bus Arbitration,” for further details.) During Read transactions, S_PAR is an input and valid for one clock after S_IRDY# assertion.

Table 3-5. Secondary PCI/PCI-X Bus Interface Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
S_PAR64	Secondary Parity, Upper 32 Bits	1	I/O PTS PCI	L19	Parity is even across S_AD[63:32] and S_CBE[7:4]#. S_PAR64 must be valid for one clock after each Address phase on transactions in which S_REQ64# is asserted. For Data phases, after S_PAR64 is valid, S_PAR64 remains valid until one Clock cycle after the current Data phase completes. Driven to a low logic level (0) when S_RSTOUT# is asserted.
S_PERR#	Secondary Parity Error	1	I/O PSTS PCI	E11	Asserted when a Data Parity error is detected for data received on the secondary interface. Before being placed into a high-impedance state, S_PERR# is driven to a de-asserted state for one cycle.
S_REQ0#	Secondary Request 0	1	If S_CFN#=0 PI otherwise OZ	B1	When the Internal PCI Arbiter is enabled (S_CFN#=0), S_REQ0# is asserted by an external device to request secondary bus ownership to perform a transaction. When using external arbitration (S_CFN#=1), S_REQ0# becomes the External Request output from the PCI 6520. If S_CFN#=0, S_REQ0# ***must*** be externally pulled up through a 10K-Ohm resistor. (Refer to Section 14.3.4, “Secondary Bus Arbitration Using External Arbiter,” for further details.)
S_REQ[7:1]#	Secondary Internal Arbiter Request 7 to 1	7	PI	E3, E4, D1, D2, D3, C1, C2	Asserted by an external device to request secondary bus ownership to perform a transaction. S_REQ[7:1]# ***must*** be externally pulled up through 10K-Ohm resistors, including those pins which are not connected to other bus master devices. S_REQ[7:1]# are not used when S_CFN#=1.

Table 3-5. Secondary PCI/PCI-X Bus Interface Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
S_REQ64#	Secondary 64-Bit Transfer Request	1	I/O PSTS PCI	C17	***Must*** be asserted low by the *central resource* (bus support functions supplied by the host system) during reset to indicate the primary bus is a 64-bit bus. Asserted with S_FRAME# by a secondary PCI Bus master to request a 64-bit Data transfer. The PCI 6520 asserts S_REQ64# low during reset and when the primary PCI master performs a 64-bit transfer or the FORCE64 option is enabled (MSCOPT[15 and/or 11]=1; PCI:46h). ***Notes:*** *If S_REQ64# is high during reset, the primary bus is a 32-bit bus, and the PCI 6520* ***must*** *park the S_AD[63:32], S_CBE[7:4]#, and S_PAR64 signals.* *If there is no central resource in the bus to assert S_REQ64# low during reset, the PCI 6520 performs a 32-bit only data transfer and ignores S_REQ64# on the bus. A three-state buffer with input pin to ground, three-state select pin to bus reset and output pin to S_REQ64# can assert S_REQ64# during reset.*
S_SERR#	Secondary System Error	1	PI	A12	S_SERR# can be driven low by any device to indicate a System error condition.
S_STOP#	Secondary Stop	1	I/O PSTS PCI	C11	Asserted by the secondary target to end the transaction on the current Data phase. Before being placed into a high-impedance state, S_STOP# is driven to a de-asserted state for one cycle.
S_TRDY#	Secondary Target Ready	1	I/O PSTS PCI	A11	Driven by the target of a transaction to indicate its ability to complete the current Data phase on the secondary bus. Before being placed into a high-impedance state, S_TRDY# is driven to a de-asserted state for one cycle.
Total		101			

Table 3-6. Clock-Related Pins

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
MSK_IN	Mask	1	PI	J1	Used with GPIO[2, 0] to shift in a serial stream of bits to the Clock Control register (CLKCNTRL; PCI:68h) to enable or disable the S_CLKO[4:0] Clock Output buffers during reset. MSK_IN can be pulled low to enable, or high to disable, all S_CLKO[4:0] buffers. ***Notes:*** *Refer to Section 4.2.2, "Secondary Clock Control," for further details.* *S_CLKOFF can also be used to enable or disable S_CLKO[4:0].*
OSCIN	External Oscillator Input	1	PI	P3	External clock input used to generate secondary output clocks when enabled through the OSCSEL# pin. Pull high or low if unused.
OSCSEL#	External Oscillator Enable	1	PI	N1	Enables external clock connection for the secondary interface. If low, the secondary bus clock outputs use the clock signal from OSCIN, instead of P_CLKIN, to generate S_CLKO[4:0]. May optionally be pulled high or low.
P_CLKIN	Primary PCI Clock Input	1	PI	N4	Provides timing for primary interface transactions.
P_CLKOE	Primary Clock Output Enable	1	PI	R5	Test pin. ***Must*** be pulled low for normal operation. Values: 0 = Disables Test function. 1 = S_CLKO3 is for primary PLL test and S_CLKO4 is for secondary PLL test.
P_CR	Primary PLL Range Control	1	PI	T6	Selects the primary PLL operating range. ***Must*** be pulled low for normal operation. Pull or tie to V_{SS}.
P_PLLEN#	Primary PLL Enable	1	PI	T7	Values: 0 = Enables primary PLL. 1 = Disables primary PLL. ***Must*** be pulled low. ***Notes:*** *The PCI-X initialization pattern and P_M66EN may also be used to automatically enable the primary PLL.* *Refer to Section 4.6, "PLL and Clock Jitter," for further details regarding use in low frequency applications.*
REFCLK	Reference Clock Input	1	PI	P2	When used, REFCLK should be a fixed frequency input (14.318 MHz recommended), which is used by the internal counters to determine the primary and secondary PCI clock frequency. Pull high or low if unused.

Table 3-6. Clock-Related Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
S_CLKIN	Secondary PCI Clock Input	1	PI	J5	Provides timing for all secondary interface transactions.
S_CLKIN_STB	Secondary Clock Input Stable	1	PI	K5	Values: 0 = S_RSTOUT# remains asserted until S_CLKIN_STB is 1. 1 = Indicates external secondary Clock PLL and external S_CLKIN are stable. Pull or tie low or high, or connect to a 3.3V power supply. If not used, connect to a 3.3V power supply.
S_CLKO0	Secondary Clock 0	1	If S_CLKOFF=0 PO otherwise PI	K3	S_CLKO0 is used as a Clock Output buffer, which is derived from the P_CLKIN or OSCIN (if OSCSEL# is low) clock; however, phase synchronization is not guaranteed. This clock can be placed into a high-impedance state, using the S_CLKOFF pin. Pull-up resistors are not required on S_CLKO0; however, a series termination resistor is required when using this pin. A pull-up resistor may be required when S_CLKO0 is used as a result of S_CLKOFF=1, or disabled (CLKCNTRL[1:0]=11b; PCI:68h), or disabled by pulling MSK_IN high. ***Note:*** *Output clocks are not recommended for PCI-X applications. External high-quality buffers are recommended.*
S_CLKO[4:1]	Secondary Clock Output 4 to 1	4	OZ	L4, L5, K1, K2	Provides P_CLKIN or OSCIN (if enabled) frequency output clocks; however, phase synchronization is not guaranteed. These clocks can be set to drive 0, using S_CLKOFF. When the S_CLKOFF pin is low, the associated Clock Control register Disable bit (CLKCNTRL[8:0]; PCI:68h) places the associated S_CLKO*x* pin into a high-impedance state. This function can be used when the disabled clock buffer is not connected to any device. Pull-up resistors are not required on S_CLKO[4:1]; however, a series termination resistor is required when using these pins. S_CLKO[4:1] may require pull-up resistors if they are disabled by pulling MSK_IN high. ***Note:*** *Output clocks are not recommended for PCI-X applications. External high-quality buffers are recommended.*

Table 3-6. Clock-Related Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
S_CLKOFF	Secondary Clock Output Disable	1	PI	K4	Values: 0 = Enables S_CLKO[4:0] output. This enable can be overridden by the Clock Control register Disable bits (CLKCNTRL[8:0]; PCI:68h). 1 = S_CLKO[4:1] output are disabled and driven low, and S_CLKO0 is placed into a high-impedance state. This disable ***cannot*** be overridden by the Clock Control register Disable bits. May optionally be pulled high or low. ***Note:*** *MSK_IN can also be used to enable or disable S_CLKO[4:0].*
S_CR	Secondary PLL Range Control	1	PI	F5	Selects the secondary PLL operating range. ***Must*** be pulled low for normal operation. Pull or tie to V_{SS}.
S_PLLEN#	Secondary PLL Enable	1	PI	E6	Values: 0 = Enables secondary PLL. 1 = Disables secondary PLL. Pull or tie low or high, or connect to a 3.3V power supply. ***Note:*** *Refer to Section 4.6, "PLL and Clock Jitter," for further details regarding use in low frequency applications.*
Total		18			

Table 3-7. Reset Pins

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
P_RSTIN#	Primary Reset Input	1	PI	L2	When asserted, outputs are asynchronously placed into a high-impedance state, P_SERR# and P_GNT# are floated, all primary PCI signals are placed into a high-impedance state, and no bus parking is asserted. All primary port PCI standard Configuration registers at offsets 00h to 3Fh revert to their default state. May require a pull-up resistor, depending on the application.
PWRGD	Power Good Input	1	PI	N3	The asserting and de-asserting edges of PWRGD can be asynchronous to P_CLKIN and S_CLKIN. ***Important Note:*** *The PCI 6520 requires a clean low-to-high transition PWRGD input. PWRGD is not internally de-bounced—it must be externally de-bounced and a high input must reflect that the power is indeed stable. When this input is low, all PCI 6520 state machines and registers are reset and all outputs, except S_RSTOUT#, are placed into a high-impedance state. Pull-up this input to 3.3V, rather than 5V.*
S_RSTOUT#	Secondary Reset Output	1	PO	H2	Asserted when either of the following conditions is met: • P_RSTIN# is asserted S_RSTOUT# remains asserted if P_RSTIN# is asserted and does ***not*** de-assert until P_RSTIN# is de-asserted. • Bridge Control register Secondary Reset bit in Configuration space is set (BCNTRL[6]=1; PCI:3Eh) S_RSTOUT# remains asserted until BCNTRL[6]=0. When asserted, places all control signals into a high-impedance state and drives S_AD[63:0], S_CBE[7:0]#, S_PAR, and S_PAR64 to a low logic level (0).
Total		3			

Table 3-8. JTAG/Boundary Scan Pins

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
TCK	Test Clock Input	1	I PU	Y2	Used to clock state information and test data into and out of the PCI 6520 during Test Access Port (TAP) operation. Pull TCK high or low to a known state, using an external resistor.
TDI	Test Data Input	1	I PU	V4	Used to serially shift test data and test instructions into the PCI 6520 during TAP operation. Pull TDI high or low to a known state, using an external resistor.
TDO	Test Data Output	1	O	W3	Used to serially shift test data and test instructions out of the PCI 6520 during TAP operation. Pull TDO high using an external resistor.
TMS	Test Mode Select	1	I PU	U5	Used to control the PCI 6520 TAP controller state. Pull TMS high or low to a known state, using an external resistor.
TRST#	Test Reset	1	I PU	Y3	Asynchronous JTAG logic reset. Provides asynchronous initialization of the TAP controller. TRST# ***must*** be externally pulled low with a 330-Ohm resistor.
Total		5			

Note: *The JTAG interface is described in Section 22, "Testability/Debug."*

Table 3-9. Serial EEPROM Pins

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
EEPCLK	Serial EEPROM Clock	1	PO	Y5	Clock signal to the serial EEPROM interface. Used during autoload and for VPD functions.
EEPDATA	Serial EEPROM Data	1	I/O	Y4	Serial data interface to the serial EEPROM. Requires an external pull-up resistor.
Total		2			

Table 3-10. General Purpose I/O Pins

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
GPIO[3:0]	General Purpose Input/ Output 3 to 0	4	I/O PU	M5, M4, M2, M1	General purpose signals, programmable as input-only or bi-directional by writing to the GPIO Output Enable register (GPIOOE[3:0]; PCI:66h). During P_RSTIN# assertion, GPIO[2, 0] are used to shift in the Clock Disable serial data. If configured as input, pull high or low, depending on the application.
GPIO[7:4]	General Purpose Input/ Output 7 to 4	4	I/O PU	A5, B5, C5, D5	General purpose signals, programmable as input-only or bi-directional by writing to the GPIO Output Enable register (GPIOOE[7:4]; PCI:9Eh). GPIO[7:4] are internally pulled up.
PCIX100MHZ	PCI-X 100 MHz	1	I/O	A3	During S_RSTOUT# assertion, PCIX100MHz pin status is latched in and used in conjunction with the S_XCAP_IN pin to generate the Secondary Clock Frequency value stored in the PCI-X Secondary Status register (PCIXSSR[8:6]; PCI:F2h). ***Notes:*** *Recommended use: 100/ 133 MHz PCI-X clock frequency.* *Pull PCIX100MHz low for all 133 MHz-capable applications. Secondary PCI-X Clock maximum speed selection:* *0 = 133 MHz.* *1 = 100 MHz.* *Refer to Table 6-3, "Secondary Clock Frequency Values (PCIXSSR[8:6]; PCI:F2h)," on page 6-51 for further details.*
Total		9			

Note: *The GPIO pins are described in Section 15, "GPIO Interface."*

Table 3-11. Miscellaneous Pins

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
BPCC_EN	Bus/Power Clock Control	1	PI	J2	When tied high and the PCI 6520 is placed into the D_{3hot} power state, the PCI 6520 places the secondary bus into the B_2 power state. The PCI 6520 disables the secondary clocks and drives them to 0. When pulled low, placing the PCI 6520 into the D_{3hot} power state has no effect on the secondary bus clocks.
DEV64#	64-Bit Device	1	PI	W4	The inverse value of this input pin is reflected in PCI-X Bridge Status register (PCIXBSR[16]; PCI:F4h). DEV64# has no effect on bus transactions. Values: 0 = 64-bit bus. 1 = 32-bit bus. May optionally be pulled high or low.
P_CLKRUN#	Primary Clock Run	1	I/O	M17	Used by the *central resource* (bus support functions supplied by the host system) to slow down or stop the PCI clock when the clock is enabled.
PRV_DEV	Private Device and Memory Enable	1	PI	J4	After power-up, the functions set by PRV_DEV can be modified by software, using the Chip Control register (CCNTRL[3:2]; PCI:40h). ***Must*** be pulled high or low. When set to 1, the PCI 6520 can mask secondary devices using IDSEL connected to S_AD[23:16] as private devices. Any Type 1 Configuration access to these IDSELs is routed to S_AD24. If there is no device on S_AD24, the re-routed Type 1 Configuration cycles are Master Aborted. The PCI 6520 also reserves Private Memory space for the secondary port. The Memory space can be programmed using the Private Memory Base and Limit registers (Base—PVTMBAR; PCI:6Ch and PVTMBARU32; PCI:70h, Limit—PVTMLMT; PCI:6Eh and PVTMLMTU32; PCI:74h). If the limit is smaller than the base, Private Memory space is disabled. The primary port ***cannot*** access this Memory space through the bridge and the secondary port does ***not*** respond to Memory cycles addressing this Private Memory space.
P_TST[1:0]	Primary Test	2	PI	T14, T15	***Reserved*** inputs. Connect P_TST[1:0] to logic 0 or 1 in layout for timing controls. (Refer to the latest reference design information.) The recommended setting is P_TST[1:0]=1, 0. Provide P_TST1 with the option of being pulled high. Provide P_TST0 with the option of being pulled low.

Table 3-11. Miscellaneous Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
P_XCAP	Primary Port PCI-X Enable	1	PI	N2	Values: 0 = Primary port protocol is set by standard PCI/PCI-X reset initialization sequence. 1 = Primary port is forced to run PCI-X protocol. If the system host bus does ***not*** issue a PCI-X Initialization Pattern, the primary port runs on PCI protocol when P_XCAP is pulled low. ***Note:*** *Refer to Section 3.2.7 for resistor requirements.*
S_CFN#	Internal Arbiter Enable	1	PI	H4	Values: 0 = Uses Internal Arbiter. 1 = Uses External Arbiter. S_REQ0# becomes REQ# output to External Arbiter and S_GNT0# becomes External Arbiter GNT# input. May optionally be pulled high or low; however, S_CFN# ***must*** be tied low to use the Internal Arbiter.
S_CLKRUN#	Secondary Clock Run	1	I/O	L18	When driven high, S_CLKRUN# slows down or stops the secondary PCI clock and is driven by a secondary PCI device to keep the clock running.
S_TST[1:0]	Secondary Test	2	PI	E14, E15	***Reserved*** inputs. Connect S_TST[1:0] to logic 0 or 1 in layout for timing controls. (Refer to the latest reference design information.) The recommended setting is S_TST[1:0]=1, 0. Provide S_TST1 with the option of being pulled high. Provide S_TST0 with the option of being pulled low.

Table 3-11. Miscellaneous Pins (Continued)

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
S_XCAP_IN	Secondary Port PCI-X Enable	1	PI	N5	S_XCAP_IN is pulled high with 56K-Ohm resistor and connected to S_XCAP_PU by way of a 1K-Ohm resistor. Together with S_XCAP_PU and PCIX100MHz, S_XCAP_IN detects the secondary interface bus mode. If S_XCAP_IN is always sampled as 1 by the PCI 6520 during secondary reset, the secondary port must run PCI-X protocol. ***Notes:*** *Refer to Section 3.2.7 for additional resistor requirements.* *Refer to Table 6-3, "Secondary Clock Frequency Values (PCIXSSR[8:6]; PCI:F2h)," on page 6-51 for further details.*
S_XCAP_PU	Secondary Port PCI-XCAP Pull-Up	1	OZ	H1	Driven low by the PCI 6520 for 32K secondary PCI clocks before de-assertion to correctly detect PCI-X Bus mode configuration. A 1K-Ohm resistor ***must*** be placed between S_XCAP_PU and S_XCAP_IN.
Total		13			

Table 3-12. Power, Ground, and No Connect Pins

Symbol	Signal Name	Total Pins	Pin Type	Pin Number	Function
P_AV_{DD}	Primary PLL Power	2	I	P6, R7	Clean +1.8V for primary PLL.
P_V_{IO}	Primary System Voltage	2	I	G1, P16	Primary bus +3.3V system signaling environment voltage. ***Note:*** *Refer to Section 3.2.8 for external resistor implementation.*
P_AV_{SS}	Primary PLL Ground	2	I	P5, R6	Clean ground for primary PLL.
S_AV_{DD}	Secondary PLL Power	2	I	F7, G6	Clean +1.8V for secondary PLL.
S_V_{IO}	Secondary System Voltage	2	I	E2, G16	Secondary bus +3.3V system signaling environment voltage. ***Note:*** *Refer to Section 3.2.8 for external resistor implementation.*
S_AV_{SS}	Secondary PLL Ground	2	I	F6, G5	Clean ground for secondary PLL.
V_{DD_CORE}	Core Power	19	I	E8, E13, F8, F13, G7, G14, H5, H6, H15, H16, N6, N15, N16, P7, P14, R8, R13, T8, T13	+1.8V supply for digital core.
V_{DD_IO}	I/O Ring Power	35	I	C3, C15, C18, F9, F10, F11, F12, F18, G8, G13, H7, H14, J6, J15, K6, K15, L3, L6, L15, M6, M15, N7, N14, P8, P13, R3, R9, R10, R11, R12, R18, V3, V6, V15, V18	+3.3V for digital I/O buffers.
V_{SS}	Ground	51	I	A1, A8, A20, C9, C12, E5, E16, G9, G10, G11, G12, J3, J7, J9, J10, J11, J12, J14, J18, K7, K9, K10, K11, K12, K14, L7, L9, L10, L11, L12, L14, M3, M7, M9, M10, M11, M12, M14, M18, P9, P10, P11, P12, T5, T16, V9, U6, V12, W5, Y1, Y20	Ground for digital core and I/O.
NC	No Connect	24	—	A2, A4, B2, B3, B4, C4, D4, E7, F14, F15, F16, G15, H3, L1, L17, M19, P15, R14, R15, R16, U7, V5, W6, Y6	No connect pins, which are not to be connected or used as routing channels. May be used in future PCI 6520 revisions.
Total		141			

4 CLOCKING

This section describes the PCI 6520 clocking requirements.

To correctly operate, the PCI 6520 requires both a primary and secondary clock.

4.1 PRIMARY AND SECONDARY CLOCK INPUTS

The PCI 6520 implements a separate clock input for each PCI interface. The primary interface is synchronized to the primary Clock input, P_CLKIN. The secondary interface is synchronized to the secondary Clock input, S_CLKIN.

The PCI 6520 primary and secondary Clock inputs can be asynchronous. There are no skew constraints between these Clock inputs; however, the maximum ratio between the primary and secondary clock frequencies are 1:4 or 4:1.

The PCI 6520 operates at a maximum frequency of 133 MHz. Output clocks S_CLKO[4:0] can be derived from P_CLKIN, P_CLKIN/2, or an external asynchronous clock source.

4.2 SECONDARY CLOCK OUTPUTS

Note: *The PCI 6520 S_CLKO[4:0] outputs are not recommended for PCI-X use. Use high-quality clock buffers for PCI-X applications.*

The PCI 6520 has five secondary Clock outputs that can be used to drive up to four external secondary bus devices. Typically, S_CLKO0 or S_CLKO4 is used to drive the PCI 6520 S_CLKIN signal.

The rules for using secondary clocks are as follows:

- Each secondary clock output is limited to no more than one PCI device. The PCI 6520 can drive one load in an embedded system or two loads when driving an adapter card slots (*that is*, the connector plus the adapter card).
- Each clock trace length, including the feedback clock to the PCI 6520 S_CLKIN signal, must have equal length and impedance.
- Terminate or disable unused secondary clock outputs to reduce power dissipation and noise in the system.

4.2.1 Disabling Secondary Clock Outputs

Secondary clock outputs may be disabled in two ways. If the S_CLKOFF input is asserted (high), S_CLKO0 is placed into a high-impedance state and S_CLKO[4:1] are disabled and driven low.

The Clock Control register (CLKCNTRL; PCI:68h) allows individual clock outputs to be disabled. Clock outputs disabled by CLKCNTRL remain disabled, regardless of S_CLKOFF status.

4.2.2 Secondary Clock Control

The PCI 6520 uses the GPIO[2, 0] pins and MSK_IN signal to input a 16-bit Serial Data stream. This data stream is shifted into the Clock Control register, as soon as P_RSTIN# is detected de-asserted and S_RSTOUT# is detected, and is used for selectively disabling S_CLKO[4:0] (CLKCNTRL[8:0]; PCI:68h). S_RSTOUT# de-assertion is delayed until the PCI 6520 completes shifting in the Clock Mask data, taking 16 Clock cycles (32 cycles if operating at 66 MHz). After that, the GPIO[2, 0] pins can be used as general purpose I/O pins.

An External Shift register should be used to load and shift the data. (Refer to Figure 4-1.) The GPIO[2, 0] pins are used for Shift register control and serial data input, which occurs by way of a dedicated input signal, MSK_IN. The Shift register circuitry is unnecessary for correct PCI 6520 operation. The Shift registers may be eliminated and, if S_CLKOFF is low, MSK_IN can be tied low to enable all S_CLKO[4:0] signals, or tied high to disable all S_CLKO[4:0] signals to three-state. If S_CLKO[4:0] are disabled to three-state, then it is recommended that pull-up resistors be added to the S_CLKO[4:0] pins to prevent noise from coupling into the PCI 6520. Table 4-1 delineates GPIO[2, 0] pin Shift register operation and Table 4-2 delineates serial data formatting, based on a design where the PCI 6520 secondary bus is used to drive up to four PCI adapter card slots or five devices in an embedded system.

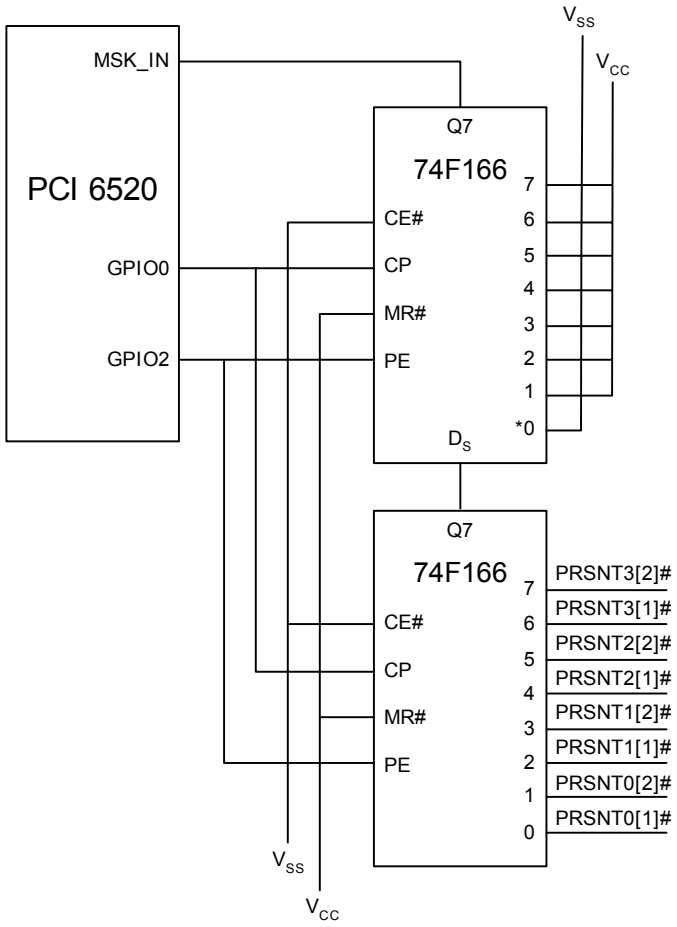


Figure 4-1. GPIO Clock Mask Implementation on System Board Example

Notes: ** Pulling the upper 74F166 bit 0 low enables S_CLKO4.*

In the Philips 74F166 PRSNTx# signals, x indicates the slot number, and the number in brackets indicates the appropriate PRSNT# signal (for example, PRSNT0[1]# is signal PRSNT1# of slot 0).

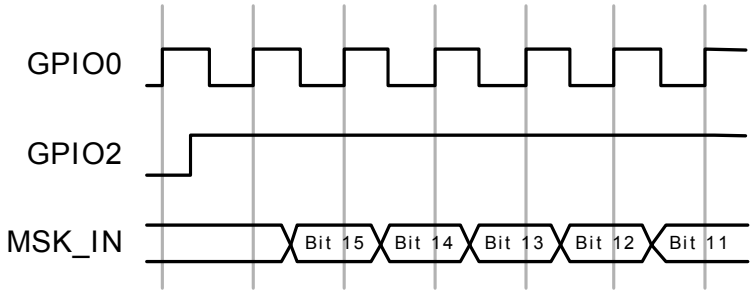


Figure 4-2. Clock Mask and Load Shift Timing

Table 4-1. GPIO Shift Register Operation

Pin	Operation
GPIO0	Shift register Clock output at 66 MHz maximum frequency.
GPIO1	***Not used.***
GPIO2	Shift Register Control. Values: 0 = Load 1 = Shift
GPIO3	***Not used.***

As noted in Table 4-2, the first eight bits contain the Philips 74F166 PRSNT*x*[2:1]# signal (refer to Figure 4-1) values for four slots, and control S_CLKO[3:0]. If one or both of the PRSNT*x*[2:1]# signals are 0, a card is present in the slot and the secondary clock for that slot is not masked. If these clocks are connected to devices and not to slots, tie one or both of the bits low, to enable the clock. The ninth bit is a clock device mask (*that is*, the bit enables or disables the clock for one device). This bit controls S_CLKO4—a value of 0 enables the clock, and 1 disables the clock.

If desired, the assignment of S_CLKO*x* to slots, devices, and PCI 6520 S_CLKIN input can be re-arranged from the assignment noted here. However, it is important that the Serial Data Stream format match the assignment of S_CLKO*x*. The GPIO[2, 0] pin serial protocol is designed to work with two Philips 74F166, 8-bit Bi-Directional Universal Shift registers.

Figure 4-1 illustrates an example application where the PCI 6520 is connected to four PCI adapter card slots. The PRSNT*x*[2:1]# pin values on the 74F166 devices are shifted into CLKCNTRL[7:0]. The PRSNT0[1]# value is shifted into CLKCNTRL[0], PRSNT0[2]# value is shifted into CLKCNTRL[1], and so forth. Bit 0 in the upper 74F166 is tied low, and thus enables S_CLKO4. In this application, S_CLKO4 may be used as the feedback to S_CLKIN.

When S_RSTOUT# is detected asserted and P_RSTIN# is detected de-asserted, the PCI 6520 drives GPIO2 low for one cycle to load the clock mask inputs into the Shift register. On the next cycle, the PCI 6520 drives GPIO2 high to perform a Shift operation. This shifts the clock mask into MSK_IN; the most significant bit is shifted in first, and the least significant bit is shifted in last. (Refer to Figure 4-2.)

After the Shift operation is complete, the PCI 6520 places GPIO[2, 0] into a high-impedance state and can de-assert S_RSTOUT# if the Secondary Reset bit is clear (BCNTRL[6]=0; PCI:3Eh). The PCI 6520 then ignores MSK_IN, and GPIO signal control reverts to the PCI 6520 GPIO Control registers. The Clock Disable bits can be subsequently modified through a Configuration Write command to the Clock Control register (CLKCNTRL) in device-specific Configuration space.

Table 4-2. GPIO Serial Data Format

CLKCNTRL[15:0]	Description	S_CLKO[4:0]
1:0	Slot 0 74F166 PRSNT0[2:1]# or Device 0	0
3:2	Slot 1 74F166 PRSNT1[2:1]# or Device 1	1
5:4	Slot 2 74F166 PRSNT2[2:1]# or Device 2	2
7:6	Slot 3 74F166 PRSNT3[2:1]# or Device 3	3
8	Device 4	4
15:9	***Reserved***	—

4.3 USING AN EXTERNAL CLOCK SOURCE

S_CLKIN_STB input allows for an indication that the secondary external clock source is stable. If this input is de-asserted (low), then this indicates that the S_CLKIN signal is not yet stable. The PCI 6520 does ***not*** de-assert S_RSTOUT# until S_CLKIN_STB is asserted (high). This ensures that a valid and stable secondary clock source is present before transactions can occur on the secondary bus.

The PCI 6520 uses two signals—OSCSEL# and OSCIN—when connecting an external clock source to the PCI 6520. During normal operation, the PCI 6520 generates S_CLKO[4:0] outputs, based on the PCI clock source (P_CLKIN). If OSCSEL# is asserted (low), then the PCI 6520 derives S_CLKO[4:0] from the OSCIN signal instead. Clock division is performed on the OSCIN and P_CLKIN clocks, depending on the P_M66EN and S_M66EN signal states.

4.4 FREQUENCY DIVISION OPTIONS

The PCI 6520 has built-in frequency division options to automatically adjust the S_CLKO[4:0] clocks for PCI 33 or 66 MHz operations. For PCI-X applications, use external, high-quality clock buffers and dividers. Table 4-3 lists the clock division ratios used, depending on the P_M66EN and S_M66EN signal states.

Table 4-3. PCI Clock Frequency Division Ratios

P_M66EN Value	S_M66EN Value	PCI Clock Frequency Division Ratio
1	1	1/1
1	0	1/2
0	1	1/1
0	0	1/1

Note: *S_M66EN **cannot** be floating.*

4.5 RUNNING SECONDARY PORT FASTER THAN PRIMARY PORT

The PCI 6520 allows the secondary port to use a higher clock frequency than that of the primary port. In this case, a secondary clock source, using an external oscillator or clock generator, must be provided.

If the external oscillator is connected to OSCIN and OSCSEL# is asserted (low), then the output generated by S_CLKO[4:0] is divided, as per Table 4-3. Division control can be disabled by pulling S_M66EN high and not connecting this pin to a PCI slot (which may be on the secondary bus). If the S_CLKO[4:0] outputs are not required, then the external clock can be fed directly into the S_CLKIN signal.

4.6 PLL AND CLOCK JITTER

The PCI 6520 uses two PLLs, one for each interface. These PLLs can be individually disabled by connecting the P_PLLEN# or S_PLLEN# pin to 1.

The minimum input frequency of each PLL is 50 MHz. If a PCI 6520 port is used in a low-speed application (*for example,* at 33 MHz), then disable the appropriate PLL by setting P_PLLEN# or S_PLLEN# to high.

For typical adapter card designs, use the adapter card's M66EN pin to control the PCI 6520's primary PLL by connecting the input of an inverter to the M66EN pin and the output to the PCI 6520's P_PLLEN# input. This ensures that the primary PLL is disabled when operating at 33 MHz. A similar method may be required to control the secondary PLL, depending on the application.

The PCI 6520 uses the PCI-X initialization pattern to automatically enable or disable the primary PLL.

PCI-X r1.0b, Table 6-1 (extrapolated in Table 4-4), defines an add-in card's capabilities according to its PCI-XCAP and M66EN signal states. The primary PLL is enabled or disabled when P_PLLEN# is pulled low, as listed in Table 4-4.

The PLL is sensitive to power and ground noise. A dedicated set of PLL Power and Ground pins are provided to reduce power and ground bounce caused by digital logic feeding into the PLLs. Connect the AV_{DD} pins for each PLL to a clean +1.8V supply and de-couple to the appropriate Ground pins. Table 4-5 details the PLL operational parameters for the primary and secondary PLLs.

Table 4-4. M66EN and PCI-XCAP Encoding

M66EN	PCI-XCAP	Conventional PCI Device Frequency Capability	PCI-X Device Frequency Capability	Primary PLL Enabled
Ground	Ground	33 MHz	Not Capable	No
	10K-Ohm Pull-Down Resistor		PCI-X 66 MHz	Yes
	Not Connected		PCI-X 133 MHz	
Not Connected	Ground	66 MHz	Not Capable	Yes
	10K-Ohm Pull-Down Resistor		PCI-X 66 MHz	
	Not Connected		PCI-X 133 MHz	

Table 4-5. PLL and Clock Jitter Parameters

Parameter	Minimum	Typical	Maximum	Unit	Condition
Input Frequency	50	—	133	MHz	—
Input Rise and Fall Time	—	—	500	ps	—
Input Cycle-to-Cycle Jitter	-100	—	+100	ps	—
Input Jitter Modulation Frequency	Must be < 100 KHz to allow PLL tracking or > 30 MHz to allow PLL filtering				
Output Cycle-to-Cycle Jitter	-150	—	+150	ps	Clean Power V_{DD} = 1.8V
Output Duty Cycle	45	—	55	%	Clean Power V_{DD} = 1.8V
Phase Lock Time	—	—	100	µs	Clean Power V_{DD} = 1.8V
PLL Power Dissipation	—	9	25	mW	Clean Power V_{DD} = 1.8V $F_{IN} = F_{OUT}$ = 133 MHz
Operating Ambient Temperature	-40	—	+85	°C	—

4.7 DETECTING PCI BUS SPEED WITH THE REFERENCE CLOCK

The PCI 6520 has a Reference Clock input, REFCLK, which is used by a timer. Any fixed-frequency source can be used as a reference clock source, although 14.318 MHz is recommended.

The timer can be set to time the primary or secondary port Clock inputs (P_CLKIN or S_CLKIN, respectively). The Timer Control register controls the count period and the PCI clock to be timed (TMRCNTRL[7:4 and 2:1]; PCI:61h, respectively). Software can then read the timer value from the Timer Counter register (TMRCNT; PCI:62h) and calculate the corresponding port's clock frequency.

4.8 PRIMARY OR SECONDARY CLOCK FREQUENCY MEASUREMENT

REFCLK is used with the Timer Control and Timer Counter registers (TMRCNTRL; PCI:61h and TMRCNT; PCI:62h, respectively) to measure the approximate bus frequency on the primary or secondary interface.

The REFCLK clock frequency should be significantly slower than that of the primary and secondary clocks.

Software must select the target bus, using bit 1 of the Timer Counter Clock Source Select bits (TMRCNTRL[2:1]; PCI:61h). Note that TMRCNTRL[2] is always cleared to 0. Software sets the Timer Enable bit to start the measurement (TMRCNTRL[0]=1; PCI:61h). TMRCNTRL[0] remains set, and software polls the Timer Stop bit until the bit is set to 1 (TMRCNTRL[3]=1; PCI:61h). When TMRCNTRL[3]=1, the measurement is complete and the result is presented in the Timer Counter register. To start a new measurement, the software must set TMRCNTRL[0] to 0, and then 1.

The measurement process counts the total measured bus clock rising edges that occurred during the overall Count Period. The counter, however, accumulates only the bus clock rising edges that occurred during the high states of each Reference Clock cycle. The Count Period (TMRCNTRL[5:4]) values indicate the number of high states used to accumulate the count (16, 32, 64, or 128). The total number of rising edges in all high states are accumulated and reported in the Timer Counter register. *For example,* if a Reference clock speed of 14.318 MHz is used, the size of each Reference clock high state is (1 / 14.318M) / 2 = 349 ns.

The Timer Counter register stores the accumulated count of rising edges within the Count Period. When the measurement is finished (indicated by TMRCNTRL[3]=1), the TMRCNT register value can be used to determine the approximate bus speed.

Example:

- Reference Clock Speed = 14.318 MHz
- Measured Clock = Secondary Bus Clock
- Count Period (Number of Windows; TMRCNTRL[5:4]=00b) = 16
- Secondary Bus Speed = 66 MHz

Based on this configuration, TMRCNTRL[1] is set to 1 because the secondary bus clock is the measured clock. TMRCNTRL[5:4] are set to 00b for 16 high states. The software first writes 0 to TMRCNTRL[0] (assuming there is a previous measurement), then writes 1 to start the measurement. Software polls TMRCNTRL[3] until the bit is set to 1.

Expected Timer Counter Count for this Example:

Because the Reference clock is 14.318 MHz, each window size is about 349 ns.

If the measured clock speed is 66 MHz, the clock period is about 15 ns. Therefore, each window count is 23 or 24 (349 / 15).

Because 16 windows (high states) were opened, the total count is in the range of 23 x 15 = 345 and 24 x 15 = 360.

Note: *For further details, refer to the TMRCNTRL and TMRCNT registers in Section 6, "Registers."*

5 RESET AND INITIALIZATION

This section describes PCI-XCAP connections and operating frequency, secondary bus mode and frequency initialization, Conventional PCI mode 66 MHz operation, reset, and register initialization.

5.1 PCI-XCAP CONNECTIONS AND OPERATING FREQUENCY

The PCI 6520 operates at up to 66 MHz in Conventional PCI mode and up to 133 MHz in PCI-X mode. The primary and secondary ports each have a PCI-XCAP input (P_XCAP and S_XCAP_IN, respectively) that determines whether it is configured as a PCI or PCI-X port. PCI-XCAP is also used to determine the operating frequency of ports operating in PCI-X mode.

5.1.1 Primary Port PCI-XCAP Connection

In a standard adapter card design, configure the P_XCAP pin as detailed in *PCI-X r1.0b*, Section 6.2.

There may be situations, however, where the standard PCI-X reset and initialization sequence is not available. *For example,* this may be the case during a CompactPCI Hot Swap device insertion.

To force the primary port to use PCI-X protocol, set the P_XCAP input to 1. If P_XCAP is set to 0, the primary port is configured to use PCI protocol and standard PCI reset and initialization occurs.

5.1.2 Secondary Port PCI-XCAP Connection

Two PCI 6520 pins are associated with the secondary PCI-XCAP connection—S_XCAP_IN and S_XCAP_PU.

The S_XCAP_IN pin is used to determine whether secondary devices are capable of PCI-X operation and at what frequency the devices can operate.

Connect S_XCAP_IN to a weak 56K-Ohm resistor, pulled up to 3.3V, and to the PCI-XCAP pin on secondary PCI-X slots. Connect S_XCAP_PU output and S_XCAP_IN together with a 1K-Ohm resistor.

During PCI-X and frequency discovery on the secondary bus, the S_XCAP_PU signal is driven low for 32K secondary PCI clocks and provides a strong pull-up resistor. This allows capacitors attached to PCI-X adapter card PCI-XCAP pins to charge. The PCI 6520 can then determine whether PCI-X cards are attached to the secondary port.

If PCI-X and frequency discovery are not required, set S_XCAP_IN to 1 to force PCI-X transactions to occur on the secondary bus, or to 0 to force Conventional PCI transactions.

5.2 SECONDARY BUS MODE AND FREQUENCY INITIALIZATION SEQUENCE

The PCI 6520 places its secondary bus in PCI-X mode based on the capabilities of devices connected to the secondary bus, independent of the primary bus mode. If only one side of a bridge is operating in PCI-X mode, the PCI 6520 translates the protocol between the two buses.

At the rising edge of P_RSTIN#, the PCI 6520 latches the frequency and mode of its primary bus. The PCI 6520 initializes the secondary bus as follows:

1. Senses the S_XCAP_IN and S_M66EN state for all devices on the secondary bus and selects the appropriate mode and clock frequency. When the secondary bus:
   - Includes one or more 33 MHz Conventional PCI devices, the bus operating frequency must be Conventional PCI 33 MHz.
   - Includes only Conventional PCI 66 MHz devices, the operating frequency is 66 MHz.
   - Includes only PCI-X 66 devices, the maximum clock frequency is 66 MHz.
   - Includes only PCI-X 100 devices, the maximum clock frequency is 100 MHz. 100 MHz selection status can be configured using the PCIX100MHZ pin.
   - Includes only PCI-X 133 devices, the maximum clock frequency is 133 MHz.
2. Asserts the appropriate signals for the PCI-X initialization pattern on the secondary bus.
3. De-asserts S_RSTOUT# to place all devices on the secondary bus in the appropriate mode.

5.3 CONVENTIONAL PCI MODE 66 MHZ OPERATION

The P_M66EN and S_M66EN signals indicate whether the primary and secondary interfaces are operating at 66 MHz. This information is needed to control the secondary bus frequency. *PCI r2.3* prohibits PCI clock frequency changes above 33 MHz, *except* during a PCI reset.

The following primary and secondary bus frequency combinations are supported when using the primary P_CLKIN signal to generate secondary clock outputs:

- 66 MHz primary bus, 66 MHz secondary bus
- 66 MHz primary bus, 33 MHz secondary bus
- 33 MHz primary bus, 33 MHz secondary bus

If P_M66EN is low (*for example*, the primary bus runs at 33 MHz), the PCI 6520 drives S_M66EN low to indicate that the secondary bus is operating at 33 MHz. If the secondary bus is set to run faster than the primary bus, S_M66EN need not be connected to secondary PCI devices.

The PCI 6520 can also generate S_CLKO[4:0] outputs from OSCIN, if enabled. When the PCI 6520 is using asynchronous clock inputs (*for example,* S_CLKO[4:0] are not derived from P_CLKIN or OSCIN), the previously listed frequencies are not the only possible clock combinations. If an external clock is used for the secondary interface, the PCI 6520 operates with a maximum ratio of 1:4 or 4:1 between the primary and secondary bus clocks.

For further details, refer to Section 4.4, “Frequency Division Options,” and Section 4.5, “Running Secondary Port Faster than Primary Port.”

5.4 RESET

This subsection describes the primary and secondary interface, and chip reset mechanisms. The PCI 6520 has two Reset inputs—PWRGD and P_RSTIN#—and one Reset output—S_RSTOUT#. In addition, the PCI 6520 can respond to Power Management-initiated and software-controlled internal resets.

After the Reset signals are de-asserted, the PCI 6520 requires 512 clocks to initialize bridge functions. During this initialization, Type 0 accesses can be accepted.

5.4.1 Power Good Reset

When PWRGD is not active, the following occurs:

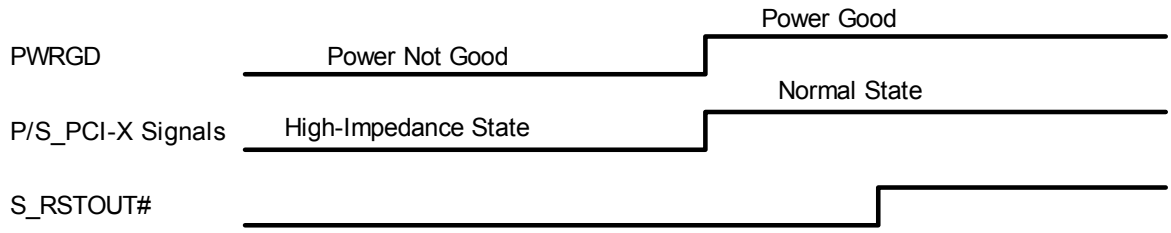
1. PCI 6520 immediately places all primary PCI-X interface signals into a high-impedance state.
2. PCI 6520 performs a full chip reset.
3. S_RSTOUT# is asserted (low).
4. All registers and extended registers with default values are reset.
5. If P_RSTIN# is low, PWRGD going from low-to-high causes the serial EEPROM to be loaded.

Note: *The PCI 6520 requires a clean low-to-high transition for PWRGD input. The PWRGD signal is not internally de-bounced—it must be externally de-bounced and a high input must reflect that the power is stable.*

The asserting and de-asserting edges of PWRGD can be asynchronous to P_CLKIN and S_CLKIN. Usually, PWRGD should not change to low when P_RSTIN# is high.

If P_RSTIN# is de-asserted before PWRGD assertion, the primary PCI-X signals remain in a high-impedance state because PWRGD is not asserted. S_RSTOUT# remains asserted until a few clocks after PWRGD assertion.

When PWRGD is de-asserted, all primary and secondary PCI signals are placed into a high-impedance state. S_RSTOUT# remains asserted until PWRGD is asserted. (Refer to Timing Diagram 5-1.)



Timing Diagram 5-1. PWRGD Assertion

5.4.2 Primary Reset Input

To properly reset, the PCI 6520 requires at least two clocks before the P_RSTIN# rising edge.

When P_RSTIN# is asserted, the following events occur:

1. PCI 6520 immediately places all primary PCI interface signals into a high-impedance state.
2. All registers are reset.
3. S_RSTOUT# is driven active to indicate a primary PCI reset.
4. P_RSTIN# assertion automatically causes a secondary port reset and S_RSTOUT# assertion. S_TRDY#, S_DEVSEL#, and S_STOP# are driven for PCI-X speed inquiry.
5. Clock Disable bits (CLKCNTRL[8:0]; PCI:68h) are shifted in at P_RSTIN# de-assertion.

The asserting and de-asserting edges of P_RSTIN# can be asynchronous to P_CLKIN and S_CLKIN.

When P_RSTIN# is asserted, all primary PCI interface signals, including the primary Request output, are immediately placed into a high-impedance state. All Posted Write and Delayed Transaction Data buffers are reset. Therefore, transactions residing in the buffers are discarded upon P_RSTIN# assertion.

5.4.3 Secondary Reset Output

The PCI 6520 is responsible for driving the secondary bus reset signal, S_RSTOUT#. The PCI 6520 asserts S_RSTOUT# when one of the following conditions is met:

- P_RSTIN# is asserted. S_RSTOUT# remains asserted when one of the following conditions is met:
  - Secondary Clock Serial Disable Mask (CLKCNTRL[8:0]; PCI:68h) is being shifted in (16 Clock cycles after P_RSTIN# de-assertion), using MSK_IN and GPIO[2, 0]
  - S_CLKIN_STB is low
- Diagnostic Control register Chip Reset and Bridge Control Secondary Reset bits are set (DCNTRL[0]=1; PCI:41h and BCNTRL[6]=1; PCI:3Eh, respectively).

  S_RSTOUT# remains asserted until the Secondary Reset Output Mask bit is cleared (DCNTRL[3]=0; PCI:41h).

When there is a D_{3hot}-to-D_0 transition with the Power Management Control/Status register Power State bits programmed to D_0 (PMCSR[1:0]=00b; PCI:E0h), S_RSTOUT# is active for 16 primary port PCI Clock cycles.

While S_RSTOUT# is asserted, S_DEVSEL#, S_STOP#, and S_TRDY# are driven for PCI-X speed inquiry. (Refer to Table 5-2 for status of all affected signals.)

5.4.4 JTAG Reset

Refer to Section 22.1.4, "JTAG Reset Input TRST#."

5.4.5 Software Resets

The Diagnostic Control register Chip Reset bit can be used to reset the PCI 6520 as the PWRGD input (DCNTRL[0]=1; PCI:41h). This action causes S_RSTOUT# assertion; however, the signals are ***not*** placed into a high-impedance state.

When the Chip Reset bit is set, all registers and chip states are reset. When chip reset completes, within four PCI Clock cycles after completion of the Configuration Write operation that sets the Chip Reset bit, the Chip Reset bit automatically clears and the PCI 6520 is ready for configuration. During chip reset, the PCI 6520 is inaccessible.

5.4.6 Power Management Internal Reset

When there is a D_{3hot}-to-D_0 transition with the Power Management Control/Status register Power State bits programmed to D_0 (PMCSR[1:0]=00b; PCI:E0h), an internal reset equivalent to P_RSTIN# is generated and all relevant registers are reset. However, S_RSTOUT# is ***not*** asserted.

5.4.7 Reset Inputs Effect on PCI 6520

Other reset controls can be used to generate secondary Reset output. Table 5-1 depicts the effect of various Reset inputs on the PCI 6520.

Table 5-1. Reset Input Effect on PCI 6520

Reset Inputs	Effect
P_RSTIN#	• Resets primary and secondary ports • Asserts S_RSTOUT# • Causes serial EEPROM load
S_CLK_STB not active	• Resets only secondary port • Asserts S_RSTOUT#
PWRGD not ready	• Asserts S_RSTOUT# • Causes serial EEPROM load
Chip Reset (DCNTRL[0]=1; PCI:41h)	Resets internal state machines
Secondary Reset (BCNTRL[6]=1; PCI:3Eh)	Asserts S_RSTOUT#

5.4.8 Pin States during PWRGD and Primary Reset

The PCI 6520 supports PWRGD and P_RSTIN#. Table 5-2 depicts the pin state for each event. With the exception of the Power Not Good state (PWRGD=0), all states assume a valid input clock.

***Legend**:*
U = Undetermined
I = Input only
T = Placed into a high-impedance state
D1 = Drive 1 to output
D0 = Drive 0 to output
D01 = Can drive both 0 or 1 to output

Table 5-2. Pin States during PWRGD and P_RSTIN#

Power-Up/Reset PCI 6520 Pins	PWRGD=0, with or without Clock, P_RSTIN#=X	PWRGD=1 and P_RSTIN#=0	Power-Up/Reset PCI 6520 Pins	PWRGD=0, with or without Clock, P_RSTIN#=X	PWRGD=1 and P_RSTIN#=0
BPCC_EN	I	I	P_V_{IO}	I	I
DEV64#	I	I	PWRGD	I	I
EEPCLK	No P_CLKIN: U With P_CLKIN: D1	D1	REFCLK	I	I
EEPDATA	No P_CLKIN: U With P_CLKIN: D1	D1	S_ACK64#	T	T
GPIO0	D1	D1	S_AD[63:0]	T	D0
GPIO[2:1]	D0	D0	S_AV_{DD}	I	I
GPIO[7:3}	T	T	S_AV_{SS}	I	I
MSK_IN	I	I	S_CBE[7:0]#	T	D0
OSCIN	I	I	S_CFN#	I	I
OSCSEL#	I	I	S_CLKIN	I	I
P_ACK64#	T	T	S_CLKIN_STB	I	I
P_AD[63:0]	T	T	S_CLKO[4:0] - S_CLKOFF=0 - S_CLKOFF=1	D01 D0	D01 D0
P_AV_{DD}	I	I	S_CLKOFF	I	I
P_AV_{SS}	I	I	S_CLKRUN#		D0
P_CBE[7:0]#	T	T	S_CR	I	I
PCIX100MHZ	T	T	S_DEVSEL#	T	T (PCI) D01 (PCI-X)
P_CLKIN	I	I	S_FRAME#	T	T

Table 5-2. Pin States during PWRGD and P_RSTIN# (Continued)

Power-Up/Reset PCI 6520 Pins	PWRGD=0, with or without Clock, P_RSTIN#=X	PWRGD=1 and P_RSTIN#=0	Power-Up/Reset PCI 6520 Pins	PWRGD=0, with or without Clock, P_RSTIN#=X	PWRGD=1 and P_RSTIN#=0
P_CLKOE	I	I	S_GNT0#	T	D1
P_CLKRUN#	I	I	S_GNT[7:1]#	T	D1
P_CR	I	I	S_IRDY#	T	T
P_DEVSEL#	T	T	S_LOCK#	T	T
P_FRAME#	T	T	S_M66EN#	I	I
P_GNT#	I	I	S_PAR	T	D0
P_IDSEL	I	I	S_PAR64	T	D0
P_IRDY#	T	T	S_PERR#	T	T
P_LOCK#	T	T	S_PLLEN#	I	I
P_M66EN#	I	I	S_REQ0#	T	I
P_PAR	T	T	S_REQ[7:1]#	I	I
P_PAR64	T	T	S_REQ64#	T	D0
P_PERR#	T	T	S_RSTOUT#	D0	D0
P_PLLEN#	I	I	S_SERR#	T	T
P_REQ#	T	T	S_STOP#	T	T (PCI) D01 (PCI-X)
P_REQ64#	T	T	S_TRDY#	T	T (PCI) D01 (PCI-X)
P_RSTIN#	I	I	S_TST[1:0]	I	I
PRV_DEV	I	I	S_V_{IO}	I	I
P_SERR#	T	T	S_XCAP_IN	I	I
P_STOP#	T	T	S_XCAP_PU	I	I
P_TRDY#	T	T	V_{DD_CORE}	I	I
P_TST[1:0]	I	I	V_{DD_IO}	I	I
P_XCAP	I	I	V_{SS}	I	I

5.5 REGISTER INITIALIZATION

The PCI 6520 Configuration registers may be initialized in one of three ways:

- Default values
- Serial EEPROM contents
- Host initialization

5.5.1 Default Initialization

After P_RSTIN# de-assertion or PWRGD assertion (whichever occurs later), the PCI 6520 automatically checks for a valid a serial EEPROM. If the serial EEPROM is not valid nor present, the PCI 6520 automatically loads default values into the Configuration registers. (Refer to the "Value after Reset" column of the register tables in Section 6, "Registers.")

5.5.2 Serial EEPROM Initialization

After P_RSTIN# de-assertion or PWRGD assertion (whichever occurs later), if the PCI 6520 finds a valid serial EEPROM, register values are loaded from the serial EEPROM and overwrite the default values. (Refer to Section 7.3, "Serial EEPROM Autoload Mode at Reset.")

5.5.3 Host Initialization

When device initialization is complete, the host system may access the appropriate registers to configure them according to system requirements.

Typically, registers are accessed by performing Type 0 Configuration accesses from the appropriate bus. The exceptions to this are the Extended registers, which are accessed using the Extended Register Index and Data registers (EXTRIDX; PCI:D3h and EXTRDATA; PCI:D4h, respectively).

For details regarding register access, refer to Section 6, "Registers."

Note: *Not all registers may be written to nor available from both sides of the bridge.*

6 REGISTERS

This section describes the PCI 6520 PCI and PCI-X registers.

As a transparent PCI-X bridge, the PCI 6520 includes the standard Type 01h Configuration Space header as defined in *P-to-P Bridge r1.2*. In Conventional PCI mode, these registers operate as defined in this specification. If either PCI 6520 interface is initialized to PCI-X mode, the requirements for these registers change to meet those of *PCI-X r1.0b*, Section 8.6.

Note: *Registers listed with a PCI offset or address are accessed by standard PCI Type 0 Configuration accesses.*

6.1 PCI CONFIGURATION REGISTER ADDRESS MAPPING

Table 6-1. PCI Configuration Register Address Mapping

PCI Configuration Register Address	To ensure software compatibility with other versions of the PCI 6520 family and to ensure compatibility with future enhancements, write 0 to all unused bits. 31 24 23 16 15 8 7 0	PCI Writable	Serial EEPROM Writable
00h	Device ID* | Vendor ID*	Yes	Yes
04h	Primary Status | Primary Command	Yes	No
08h	Class Code* | Revision ID	Yes	Yes
0Ch	Built-In Self-Test ***(Not Supported)*** | Header Type* | Primary Latency Timer | Cache Line Size	Yes	Yes
10h – 14h	***Reserved***	No	No
18h	Secondary Latency Timer | Subordinate Bus Number | Secondary Bus Number | Primary Bus Number	Yes	No
1Ch	Secondary Status | I/O Limit | I/O Base	Yes	No
20h	Memory Limit | Memory Base	Yes	No
24h	Prefetchable Memory Limit | Prefetchable Memory Base	Yes	No
28h	Prefetchable Memory Base Upper 32 Bits	Yes	No
2Ch	Prefetchable Memory Limit Upper 32 Bits	Yes	No
30h	I/O Limit Upper 16 Bits | I/O Base Upper 16 Bits	Yes	No
34h	***Reserved*** | New Capability Pointer	No	No
38h	***Reserved***	No	No
3Ch	Bridge Control | Interrupt Pin | ***Reserved***	Yes	No
40h	Arbiter Control | Diagnostic Control | Chip Control	Yes	No
44h	Miscellaneous Options | Timeout Control | Primary Flow-Through Control	Yes	Yes
48h	Secondary Incremental Prefetch Count | Primary Incremental Prefetch Count | Secondary Initial Prefetch Count | Primary Initial Prefetch Count	Yes	Yes
4Ch	Buffer Control | Secondary Flow-Through Control | Secondary Maximum Prefetch Count | Primary Maximum Prefetch Count	Yes	Yes
50h	***Reserved*** | Test | Internal Arbiter Control	Yes	No
54h	Serial EEPROM Data | Serial EEPROM Address | Serial EEPROM Control	Yes	No
58h	***Reserved***	No	No
5Ch	***Reserved***	No	No
60h	Timer Counter | Timer Control | ***Reserved***	Yes	No
64h	GPIO[3:0] Input Data | GPIO[3:0] Output Enable | GPIO[3:0] Output Data | P_SERR# Event Disable	Yes	No
68h	Clock Run | P_SERR# Status | Clock Control	Yes	No
6Ch	Private Memory Limit | Private Memory Base	Yes	No
70h	Private Memory Base Upper 32 Bits	Yes	No
74h	Private Memory Limit Upper 32 Bits	Yes	No
78h – 98h	***Reserved***	No	No

Table 6-1. PCI Configuration Register Address Mapping (Continued)

PCI Configuration Register Address	To ensure software compatibility with other versions of the PCI 6520 family and to ensure compatibility with future enhancements, write 0 to all unused bits. 31 24 / 23 16 / 15 8 / 7 0	PCI Writable	Serial EEPROM Writable
9Ch	GPIO[7:4] Input Data | GPIO[7:4] Output Enable | GPIO[7:4] Output Data | Read-Only Register Control	Yes	No
A0h	***Reserved***	No	No
ACh – CCh	***Reserved***	No	No
D0h	Extended Register Index | ***Reserved***	Yes	No
D4h	Extended Register Data	Yes	No
D8h	***Reserved***	No	No
DCh	Power Management Capabilities | Power Management Next Capability Pointer (E8h) | Power Management Capability ID (01h)	Yes	Yes
E0h	Power Management Data* | PMCSR Bridge Supports Extensions | Power Management Control/Status*	Yes	Yes
E4h	***Reserved***	No	No
E8h	VPD Address (0h) | VPD Next Capability Pointer (F0h) | VPD Capability ID (03h)	Yes	No
ECh	VPD Data (0h)	Yes	No
F0h	PCI-X Secondary Status | PCI-X Next Capability Pointer (0h) | PCI-X Capability ID (07h)	Yes	No
F4h	PCI-X Bridge Status	Yes	No
F8h	PCI-X Upstream Split Transaction	Yes	No
FCh	PCI-X Downstream Split Transaction	Yes	No

Notes: ** Writable only when the Read-Only Registers Write Enable bit is set (RRC[7]=1; PCI:9Ch). Refer to the individual register descriptions to determine which bits are writable.*

Refer to the individual register descriptions to determine which bits are writable.

6.1.1 PCI Type 1 Header

Register 6-1. (PCIIDR; PCI:00h) PCI Configuration ID

Bit	Description	Read	Write	Value after Reset
15:0	**Vendor ID.** Identifies PCI 6520 manufacturer. Defaults to the PCI-SIG-issued PLX Vendor ID (10b5h), if a blank or no serial EEPROM is present.	Yes	Only if RRC[7]=1; Serial EEPROM	10b5h
31:16	**Device ID.** Identifies the particular device. Defaults to PLX PCI 6520 part number (6520h), if a blank or no serial EEPROM is present.	Yes	Only if RRC[7]=1; Serial EEPROM	6520h

Register 6-2. (PCICR; PCI:04h) Primary PCI Command

Bit	Description	Read	Write	Value after Reset
0	**I/O Space Enable.** Controls bridge response to I/O accesses on primary interface. Values: 0 = Ignores I/O transactions 1 = Enables response to I/O transactions	Yes	Yes	0
1	**Memory Space Enable.** Controls bridge response to Memory accesses on primary interface. Values: 0 = Ignores Memory transactions 1 = Enables response to Memory transactions	Yes	Yes	0
2	**Bus Master Enable.** Controls bridge ability to operate as a master on primary interface. Values: 0 = Does not initiate transactions on primary interface and disables response to Memory or I/O transactions on secondary interface 1 = Enables bridge to operate as a master on primary interface	Yes	Yes	0
3	**Special Cycle Enable.** ***Not Supported.***	Yes	No	0
4	**Memory Write and Invalidate Enable.** ***Not Supported.***	Yes	No	0
5	**VGA Palette Snoop Enable.** Controls bridge response to VGA-compatible Palette accesses. Values: 0 = Ignores VGA Palette accesses on primary interface 1 = Enables response to VGA Palette writes on primary interface (I/O address AD[9:0]=3C6h, 3C8h, and 3C9h) ***Note:*** *If BCNTRL[3]=1; PCI:3Eh (VGA Enable bit), then VGA Palette accesses are forwarded, regardless of the PCICR[5] value.*	Yes	Yes	0
6	**Parity Error Response Enable.** Controls bridge response to Parity errors. Values: 0 = Ignores Parity errors 1 = Performs normal parity checking	Yes	Yes	0
7	**Wait Cycle Control.** If set to 1, the PCI 6520 performs address/data stepping.	Yes	Yes	1

Register 6-2. (PCICR; PCI:04h) Primary PCI Command (Continued)

Bit	Description	Read	Write	Value after Reset
8	**P_SERR# Enable.** Controls the primary System Error (P_SERR#) pin enable. Values: 0 = Disables P_SERR# driver 1 = Enables P_SERR# driver	Yes	Yes	0
9	**Fast Back-to-Back Enable.** Controls bridge ability to generate Fast Back-to-Back transactions to various devices on primary interface. Values: 0 = No Fast Back-to-Back transactions 1 = ***Reserved***; PCI 6520 does not generate Fast Back-to-Back cycles	Yes	Yes	0
15:10	***Reserved.***	Yes	No	0h

Register 6-3. (PCISR; PCI:06h) Primary PCI Status

Bit	Description	Read	Write	Value after Reset
3:0	***Reserved.***	Yes	No	0h
4	**New Capability Functions Support.** Writing 1 supports New Capabilities Functions. The New Capability Function ID is located at the PCI Configuration space offset, determined by the New Capabilities linked list pointer value at CAP_PTR; PCI:34h.	Yes	No	1
5	**66 MHz-Capable.** If set to 1, this device supports a 66 MHz PCI clock environment.	Yes	No	1
6	**UDF.** No User-Definable Features.	Yes	No	0
7	**Fast Back-to-Back Capable.** Fast Back-to-Back write capable on primary port.	Yes	No	0
8	**Data Parity Error Detected.** Set when the following conditions are met: • P_PERR# is asserted, and • Command register Parity Error Response Enable bit is set (PCICR[6]=1; PCI:04h) Writing 1 clears bit to 0.	Yes	Yes/Clr	0
10:9	**DEVSEL# Timing**. Reads as 01b to indicate PCI 6520 responds no slower than with medium timing.	Yes	No	01b
11	**Signaled Target Abort.** Set by a target device when a Target Abort cycle occurs. Writing 1 clears bit to 0.	Yes	Yes/Clr	0
12	**Received Target Abort.** Set to 1 by the PCI 6520 when transactions are terminated with Target Abort. Writing 1 clears bit to 0.	Yes	Yes/Clr	0
13	**Received Master Abort.** Set to 1 by the PCI 6520 when transactions are terminated with Master Abort. Writing 1 clears bit to 0.	Yes	Yes/Clr	0
14	**Signaled System Error.** Set when P_SERR# is asserted. Writing 1 clears bit to 0.	Yes	Yes/Clr	0
15	**Parity Error Detected.** Set when a Parity error is detected, regardless of the Parity Error Response Enable bit state (PCICR[6]=x; PCI:04h). Writing 1 clears bit to 0.	Yes	Yes/Clr	0

Register 6-4. (PCIREV; PCI:08h) PCI Revision ID

Bit	Description	Read	Write	Value after Reset
7:0	**Revision ID.** PCI 6520 revision.	Yes	No	CBh

Register 6-5. (PCICCR; PCI:09h – 0Bh) PCI Class Code

Bit	Description	Read	Write	Value after Reset
7:0	**Register Level Programming Interface.** None defined.	Yes	Only if RRC[7]=1; Serial EEPROM	0h
15:8	**Subclass Code.** PCI-to-PCI bridge or other bridge device.	Yes	Only if RRC[7]=1; Serial EEPROM	04h
23:16	**Base Class Code.** Bridge device.	Yes	Only if RRC[7]=1; Serial EEPROM	06h

Register 6-6. (PCICLSR; PCI:0Ch) PCI Cache Line Size

Bit	Description	Read	Write	Value after Reset
7:0	**System Cache Line Size.** Specified in units of 32-bit words (Dwords). Only cache line sizes of a power of two are valid. Maximum value is 20h. For values greater than 20h, PCI 6520 operates as if PCICLSR is programmed with value of 08h. Used when terminating Memory Write and Invalidate transactions. Memory Read prefetching is controlled by the Prefetch Count registers.	Yes	Yes	0h

Register 6-7. (PCILTR; PCI:0Dh) Primary PCI Bus Latency Timer

Bit	Description	Read	Write	Value after Reset
7:0	**Primary PCI Bus Latency Timer.** Specifies amount of time (in units of PCI Bus clocks) the PCI 6520, as a bus master, can burst data on the primary PCI Bus.	Yes	Yes	0h

Register 6-8. (PCIHTR; PCI:0Eh) PCI Header Type

Bit	Description	Read	Write	Value after Reset
6:0	**Configuration Layout Type.** Specifies register layout at offsets 10h to 3Fh in Configuration space. Header Type 0 is defined for PCI devices other than PCI-to-PCI bridges (Header Type 1) and Cardbus bridges (Header Type 2).	Yes	Only if RRC[7]=1; Serial EEPROM	1h
7	**Multi-Function Device.** Value of 1 indicates multiple (up to eight) functions (logical devices), each containing its own, individually addressable Configuration space, 64 Dwords in size.	Yes	Only if RRC[7]=1; Serial EEPROM	0

Register 6-9. (PCIBISTR; PCI:0Fh) PCI Built-In Self-Test

Bit	Description	Read	Write	Value after Reset
7:0	**Built-In Self-Test (BIST).** ***Not Supported.***	Yes	No	0h

Register 6-10. (PCIPBNO; PCI:18h) PCI Primary Bus Number

Bit	Description	Read	Write	Value after Reset
7:0	**Primary Bus Number.** Programmed with the PCI and/or PCI-X Bus number to which the primary bridge interface is connected.	Yes	Yes	0h

Register 6-11. (PCISBNO; PCI:19h) PCI Secondary Bus Number

Bit	Description	Read	Write	Value after Reset
7:0	**Secondary Bus Number.** Programmed with the PCI and/or PCI-X Bus number to which the secondary bridge interface is connected.	Yes	Yes	0h

Register 6-12. (PCISUBNO; PCI:1Ah) PCI Subordinate Bus Number

Bit	Description	Read	Write	Value after Reset
7:0	**Subordinate Bus Number.** Programmed with the PCI and/or PCI-X Bus Number with the highest number subordinate to the bridge.	Yes	Yes	0h

Register 6-13. (PCISLTR; PCI:1Bh) Secondary PCI Bus Latency Timer

Bit	Description	Read	Write	Value after Reset
7:0	**Secondary PCI Bus Latency Timer.** Specifies the amount of time (in units of PCI Bus clocks) the PCI 6520, as a bus master, can burst data on the secondary PCI Bus.	Yes	Yes	0h

Register 6-14. (PCIIOBAR; PCI:1Ch) I/O Base

Bit	Description	Read	Write	Value after Reset
7:0	**I/O Base.** Specifies the I/O Base Address Range bits [15:12] for forwarding the cycle through the bridge (Base Address bits [11:0] are assumed to be 0h). Used in conjunction with the I/O Limit, I/O Base Upper 16 Bits, and I/O Limit Upper 16 Bits registers (PCIIOLMT; PCI:1Dh, PCIIOBARU16; PCI:30h, and PCIIOLMTU16; PCI:32h, respectively) to specify a range of 32-bit addresses supported for PCI Bus I/O transactions. The lower four bits [3:0] are Read-Only and hardcoded to 0001b to indicate 32-bit I/O addressing support.	Yes	Yes [7:4]	1h

Register 6-15. (PCIIOLMT; PCI:1Dh) I/O Limit

Bit	Description	Read	Write	Value after Reset
7:0	**I/O Limit.** Specifies the Upper I/O Limit Address Range bits [15:12] for forwarding the cycle through the bridge (Limit Address bits [11:0] are assumed to be FFFh). Used in conjunction with the I/O Base, I/O Base Upper 16 Bits, and I/O Limit Upper 16 Bits registers (PCIIOBAR; PCI:1Ch, PCIIOBARU16; PCI:30h, and PCIIOLMTU16; PCI:32h, respectively) to specify a range of 32-bit addresses supported for PCI Bus I/O transactions. The lower four bits [3:0] are Read-Only and hardcoded to 0001b to indicate 32-bit I/O addressing support.	Yes	Yes [7:4]	1h

Register 6-16. (PCISSR; PCI:1Eh) Secondary PCI Status

Bit	Description	Read	Write	Value after Reset
4:0	***Reserved.***	Yes	No	0h
5	**66 MHz-Capable.** If set to 1, the PCI 6520 supports a 66 MHz PCI clock environment.	Yes	No	1
6	**UDF.** No User-definable features.	Yes	No	0
7	**Fast Back-to-Back Capable.** Fast Back-to-Back write capable on secondary port. ***Not Supported.***	Yes	No	0
8	**Data Parity Error Detected.** Set when the following conditions are met: • S_PERR# is asserted, and • Command register Parity Error Response Enable bit is set (PCICR[6]=1; PCI:04h) Writing 1 clears bit to 0.	Yes	Yes/Clr	0
10:9	**DEVSEL# Timing**. Reads as 01b to indicate PCI 6520 responds no slower than with medium timing.	Yes	No	01b
11	**Signaled Target Abort.** Set by a target device when a Target Abort cycle occurs. Writing 1 clears bit to 0.	Yes	Yes/Clr	0
12	**Received Target Abort.** Set to 1 by PCI 6520 when transactions are terminated with Target Abort. Writing 1 clears bit to 0.	Yes	Yes/Clr	0
13	**Received Master Abort.** Set to 1 by PCI 6520 when transactions are terminated with Master Abort. Writing 1 clears bit to 0.	Yes	Yes/Clr	0
14	**Signaled System Error.** Set when S_SERR# is asserted. Writing 1 clears bit to 0.	Yes	Yes/Clr	0
15	**Parity Error Detected.** Set when a Parity error is detected, regardless of the Parity Error Response Enable bit state (PCICR[6]=x; PCI:04h). Writing 1 clears bit to 0.	Yes	Yes/Clr	0

Register 6-17. (PCIMBAR; PCI:20h) Memory Base

Bit	Description	Read	Write	Value after Reset
15:0	**Memory Base.** Specifies the Memory-Mapped I/O Base Address Range bits [31:20] for forwarding the cycle through the bridge. The upper 12 bits corresponding to [31:20] are writable. The lower 20 Address bits [19:0] are assumed to be 0h. Used in conjunction with the Memory Limit register (PCIMLMT; PCI:22h) to specify a range of 32-bit addresses supported for PCI Bus Memory-Mapped I/O transactions. The lower four bits [3:0] are Read-Only and hardcoded to 0h.	Yes	Yes [15:4]	0h

Register 6-18. (PCIMLMT; PCI:22h) Memory Limit

Bit	Description	Read	Write	Value after Reset
15:0	**Memory Limit.** Specifies the Upper Memory-Mapped I/O Limit Address Range bits [31:20] for forwarding the cycle through the bridge. The upper 12 bits corresponding to [31:20] are writable. The lower 20 Address bits [19:0] are assumed to be F_FFFFh. Used in conjunction with the Memory Base register (PCIMBAR; PCI:20h) to specify a range of 32-bit addresses supported for PCI Bus Memory-Mapped I/O transactions. The lower four bits [3:0] are Read-Only and hardcoded to 0h.	Yes	Yes [15:4]	0h

Register 6-19. (PCIPMBAR; PCI:24h) Prefetchable Memory Base

Bit	Description	Read	Write	Value after Reset
15:0	**Prefetchable Memory Base.** Specifies the Prefetchable Memory-Mapped Base Address Range bits [31:20] for forwarding the cycle through the bridge. The upper 12 bits corresponding to [31:20] are writable. The lower 20 Address bits [19:0] are assumed to be 0h. Used in conjunction with the Prefetchable Memory Limit, Prefetchable Memory Base Upper 32 Bits, and Prefetchable Memory Limit Upper 32 Bits registers (PCIPMLMT; PCI:26h, PCIPMBARU32; PCI:28h, and PCIPMLMTU32; PCI:2Ch, respectively) to specify a range of 64-bit addresses supported for Prefetchable Memory transactions on the PCI Bus. The lower four Read-Only bits are hardcoded to 01h, indicating 64-bit address support.	Yes	Yes [15:4]	1h

Register 6-20. (PCIPMLMT; PCI:26h) Prefetchable Memory Limit

Bit	Description	Read	Write	Value after Reset
15:0	**Prefetchable Memory Limit.** Specifies the Upper Prefetchable Memory-Mapped Limit Address Range bits [31:20] for forwarding the cycle through the bridge. The lower 20 Address bits [19:0] are assumed to be F_FFFFh. Used in conjunction with the Prefetchable Memory Base, Prefetchable Memory Base Upper 32 Bits, and Prefetchable Memory Limit Upper 32 Bits registers (PCIPMBAR; PCI:24h, PCIPMBARU32; PCI:28h, and PCIPMLMTU32; PCI:2Ch, respectively) to specify a range of 64-bit addresses supported for Prefetchable Memory transactions on the PCI Bus. The lower four Read-Only bits are hardcoded to 01h, indicating 64-bit address support.	Yes	Yes [15:4]	1h

Register 6-21. (PCIPMBARU32; PCI:28h) Prefetchable Memory Base Upper 32 Bits

Bit	Description	Read	Write	Value after Reset
31:0	**Prefetchable Memory Base Upper 32 Bits.** Specifies the Upper Prefetchable Memory-Mapped Base Address Range bits [63:32] for forwarding the cycle through the bridge. The lower 20 Address bits [19:0] are assumed to be 0h. Used in conjunction with the Prefetchable Memory Base, Prefetchable Memory Limit, and Prefetchable Memory Limit Upper 32 Bits registers (PCIPMBAR; PCI:24h, PCIPMLMT; PCI:26h, and PCIPMLMTU32; PCI:2Ch, respectively) to specify a range of 64-bit addresses supported for Prefetchable Memory transactions on the PCI Bus.	Yes	Yes	0h

Register 6-22. (PCIPMLMTU32; PCI:2Ch) Prefetchable Memory Limit Upper 32 Bits

Bit	Description	Read	Write	Value after Reset
31:0	**Prefetchable Memory Limit Upper 32 Bits.** Specifies the Upper Prefetchable Memory-Mapped Limit Address Range bits [63:32] for forwarding the cycle through the bridge. The lower 20 Address bits [19:0] are assumed to be F_FFFFh. Used in conjunction with the Prefetchable Memory Base, Prefetchable Memory Limit, and Prefetchable Memory Base Upper 32 Bits registers (PCIPMBAR; PCI:24h, PCIPMLMT; PCI:26h, and PCIPMBARU32; PCI:28h, respectively) to specify a range of 64-bit addresses supported for Prefetchable Memory transactions on the PCI Bus.	Yes	Yes	0h

Register 6-23. (PCIIOBARU16; PCI:30h) I/O Base Upper 16 Bits

Bit	Description	Read	Write	Value after Reset
15:0	**I/O Base Upper 16 Bits.** Specifies the Upper I/O Base Address Range bits [31:16] for forwarding the cycle through the bridge. Base Address bits [11:0] are assumed to be 0h. Used in conjunction with the I/O Base, I/O Limit, and I/O Limit Upper 16 Bits registers (PCIIOBAR; PCI:1Ch, PCIIOLMT; PCI:1Dh, and PCIIOLMTU16; PCI:32h, respectively) to specify a range of 32-bit addresses supported for PCI Bus I/O transactions.	Yes	Yes	0h

Register 6-24. (PCIIOLMTU16; PCI:32h) I/O Limit Upper 16 Bits

Bit	Description	Read	Write	Value after Reset
15:0	**I/O Limit Upper 16 Bits.** Specifies the Upper I/O Limit Address Range bits [31:16] for forwarding the cycle through the bridge. Limit Address bits [11:0] are assumed to be FFFh. Used in conjunction with the I/O Base, I/O Limit, and I/O Base Upper 16 Bits registers (PCIIOBAR; PCI:1Ch, PCIIOLMT; PCI:1Dh, and PCIIOBARU16; PCI:30h, respectively) to specify a range of 32-bit addresses supported for PCI Bus I/O transactions.	Yes	Yes	0h

Register 6-25. (CAP_PTR; PCI:34h) New Capability Pointer

Bit	Description	Read	Write	Value after Reset
7:0	**New Capability Pointer.** Provides an offset into PCI Configuration space for the Power Management capability location in the New Capabilities Linked List.	Yes	No	DCh
31:8	***Reserved.***	Yes	No	0h

Register 6-26. (PCIIPR; PCI:3Dh) PCI Interrupt Pin

Bit	Description	Read	Write	Value after Reset
7:0	**Interrupt Pin.** Reads as 0h to indicate that PCI 6520 does not use interrupt pins.	Yes	No	0h

Register 6-27. (BCNTRL; PCI:3Eh) Bridge Control

Bit	Description	Read	Write	Value after Reset
0	**Parity Error Response Enable.** Controls bridge response to Parity errors on secondary interface. Values: 0 = Ignores Address and Data Parity errors on secondary interface 1 = Enables Parity error reporting and detection on secondary interface	Yes	Yes	0
1	**S_SERR# Enable.** Controls forwarding of S_SERR# to primary interface. Values: 0 = Disables S_SERR# forwarding to primary 1 = Enables S_SERR# forwarding to primary	Yes	Yes	0
2	**ISA Enable.** Controls bridge response to ISA I/O addresses, which is limited to the first 64 KB. Values: 0 = Forwards I/O addresses in the range defined by the I/O Base and Limit registers (PCIIOBAR; PCI:1Ch and PCIIOLMT; PCI:1Dh, respectively). 1 = Blocks forwarding of ISA I/O addresses in the range defined by the I/O Base and Limit registers in the first 64 KB of I/O space that address the last 768 bytes in each 1-KB block. Secondary I/O transactions are forwarded upstream, if the address falls within the last 768 bytes in each 1-KB block. Command Configuration register Master Enable bit must also be set (PCICR[2]=1; PCI:04h) to enable ISA.	Yes	Yes	0
3	**VGA Enable.** Controls bridge response to VGA-compatible addresses. Values: 0 = Does not forward VGA-compatible Memory nor I/O addresses from primary to secondary 1 = Forwards VGA-compatible Memory and I/O addresses from primary to secondary, regardless of other settings ***Note:*** *If set to 1, then I/O addresses in the range of 3B0h to 3BBh and 3C0h to 3DFh are forwarded, regardless of the PCICR[5]; PCI:04h or BCNTRL[2] values.*	Yes	Yes	0

Register 6-27. (BCNTRL; PCI:3Eh) Bridge Control (Continued)

Bit	Description	Read	Write	Value after Reset
4	***Reserved.***	Yes	No	0
5	**Master Abort Mode.** Controls bridge behavior in response to Master Aborts on secondary interface. Values: 0 = Does not report Master Aborts (return FFFF_FFFFh on reads or discard data on writes). 1 = Reports Master Aborts by signaling Target Abort. If the Master Abort is the result of a primary-to-secondary Posted Write cycle, P_SERR# is asserted (PCICR[8]=1; PCI:04h). ***Note:*** *During Lock cycles, PCI 6520 ignores this bit, and completes the cycle as a Target Abort.*	Yes	Yes	0
6	**Secondary Reset.** Forces S_RSTOUT# assertion on secondary interface. Values: 0 = Does not force S_RSTOUT# assertion 1 = Forces S_RSTOUT# assertion	Yes	Yes	0
7	**Fast Back-to-Back Enable.** Controls bridge ability to generate Fast Back-to-Back transactions to various devices on secondary interface. Values: 0 = No Fast Back-to-Back transaction 1 = ***Reserved***; PCI 6520 does not generate Fast Back-to-Back cycles	Yes	Yes	0
8	**Primary Master Timeout (Discard Timer).** Sets the maximum number of PCI clocks for an initiator on the primary bus to repeat the Delayed transaction request. Values: 0 = Timeout after 2^{15} PCI clocks 1 = Timeout after 2^{10} PCI clocks	Yes	Yes	0
9	**Secondary Master Timeout (Discard Timer).** Sets the maximum number of PCI clocks for an initiator on the secondary bus to repeat the Delayed transaction request. Values: 0 = Timeout after 2^{15} PCI clocks 1 = Timeout after 2^{10} PCI clocks	Yes	Yes	0
10	**Master Timeout Status.** Set to 1 when primary or secondary Master Timeout occurs. Writing 1 clears bit to 0.	Yes	Yes/Clr	0
11	**Master Timeout P_SERR# Enable.** Enable P_SERR# assertion during Master Timeout. Values: 0 = P_SERR# not asserted on Master Timeout 1 = P_SERR# asserted on primary or secondary Master Timeout	Yes	Yes	0
15:12	***Reserved.***	Yes	No	0h

6.1.2 Device-Specific

6.1.2.1 Chip, Diagnostic, and Arbiter Control

Register 6-28. (CCNTRL; PCI:40h) Chip Control

Bit	Description	Read	Write	Value after Reset
0	***Reserved.***	Yes	No	0
1	**Memory Write Disconnect Control.** Controls when PCI 6520, as a target, Disconnects Memory transactions. Values: 0 = Disconnects on queue full or on a 4-KB boundary 1 = Disconnects on a Cache Line boundary, when the queue fills, or on a 4-KB boundary	Yes	Yes	0
2	**Private Memory Enable.** The Memory space can be programmed using the Private Memory Base and Limit registers (PVTMBAR; PCI:6Ch and PVTLMT; PCI:6Eh, respectively). If the Limit is smaller than the Base, the Private Memory space is disabled regardless of bit setting. When enabled, the primary port ***cannot*** access primary and secondary Private Memory space through the bridge and the secondary port does ***not*** respond to Memory cycles addressing the Private Memory space. Resets to the value presented on the PRV_DEV input pin. After reset, bit can be reprogrammed. Values: 0 = Disables Private Memory block 1 = Enables Private Memory block	Yes	Yes	PRV_DEV
3	**Private Device Enable.** PCI 6520 can re-route secondary IDSELs using S_AD[23:16] for private devices. A Type 1 Configuration access on the primary bus (which would normally result in the assertion of an IDSEL connected to S_AD[23:16]) is routed to S_AD24. If there is no device on S_AD24, the re-routed Type 1 Configuration cycles result in a Master Abort. Re-routing allows S_AD[23:16] to be used for secondary private devices. Resets to the value presented in the PRV_DEV input pin. After reset, bit can be reprogrammed. Values: 0 = Does not re-route IDSEL assertions 1 = Enables the re-routing of the secondary IDSEL S_AD[23:16] to S_AD24	Yes	Yes	PRV_DEV
4	**Secondary Bus Prefetch Disable.** Controls PCI 6520 ability to prefetch during upstream Memory Read transactions. Values: 0 = Prefetches and does not forward Byte Enables during Memory Read transactions. 1 = Requests only 1 Dword from the target during Memory Read transactions and forwards Byte Enables. PCI 6520 returns a Target Disconnect to the requesting master on the first Data transfer. Memory Read Line and Memory Read Multiple transactions remain prefetchable.	Yes	Yes	0
7:5	***Reserved.***	Yes	No	000b

Register 6-29. (DCNTRL; PCI:41h) Diagnostic Control

Bit	Description	Read	Write	Value after Reset
0	**Chip Reset.** Chip and secondary bus reset. Setting bit activates full chip reset, asserts S_RSTOUT#, and forces the Bridge Control register Secondary Reset bit to be set (BCNTRL[6]=1; PCI:3Eh). After resetting the PCI 6520 registers, bit is cleared; however, BCNTRL[6] remains set to 1. Writing 0 has no effect.	Yes	Yes	0
2:1	***Reserved and must be set to 00b.***	Yes	Yes	00b
3	**Secondary Reset Output Mask.** ***Not Supported***.	Yes	No	0
7:4	***Reserved.***	Yes	No	0h

Register 6-30. (ACNTRL; PCI:42h) Arbiter Control

Bit	Description	Read	Write	Value after Reset
7:0	**Arbiter Control.** Each bit controls whether a secondary bus master is assigned to the high- or low-priority group. Bits correspond to request inputs S_REQ[7:0]#, respectively. Value of 1h assigns the bus master to the high-priority group.	Yes	Yes	0h
8	***Reserved.***	Yes	Yes	0
9	**PCI 6520 Priority.** Defines whether PCI 6520 secondary port is in the high- or low-priority group. 0 = Low-priority group 1 = High-priority group	Yes	Yes	1
11:10	***Reserved.***	Yes	No	00b
12	**Primary Port Ordering Rule.** ***Reserved and must be set to 0.***	Yes	Yes	0
13	**Secondary Port Ordering Rule**. ***Reserved and must be set to 0.***	Yes	Yes	0
14	**Upstream 64-Bit Cycle Control.** ***Reserved and must be set to 0.***	Yes	Yes	0
15	**Downstream 64-Bit Cycle Control.** ***Reserved and must be set to 0.***	Yes	Yes	0

6.1.2.2 Primary Flow-Through Control

Register 6-31. (PFTCR; PCI:44h) Primary Flow-Through Control

Bit	Description	Read	Write	Value after Reset
2:0	**Primary Posted Write Completion Wait Count.** Maximum number of clocks the PCI 6520 waits for Posted Write data from the initiator, if delivering Write data in Flow-Through mode, with the Internal Post Write queues almost empty. If the count is exceeded, without additional data from the initiator, the cycle to the target is terminated and later completed. Value: 000b = De-asserts S_IRDY# and waits seven clocks for data on the primary bus before terminating cycle All other values are ***Reserved.*** ***Note:*** *To specify other clock values, use software or the serial EEPROM to set these bits to any value between two and seven clocks.*	Yes	Yes; Serial EEPROM	000b
3	***Reserved.***	Yes	No	0
6:4	**Primary Delayed Read Completion Wait Count.** Maximum number of clocks the PCI 6520 waits for Delayed Read data from the target if returning Read data in Flow-Through mode, and the Internal Delayed Read queue is almost full. If the count is exceeded without additional space in the queue, the cycle to the target is terminated, and completed when the initiator Retries the remainder of the cycle. Value: 000b = De-asserts S_IRDY# and waits seven clocks for further data to be transferred to the primary bus before terminating cycle All other values are ***Reserved.*** ***Note:*** *To specify other clock values, use software or the serial EEPROM to set these bits to any value between two and seven clocks.*	Yes	Yes; Serial EEPROM	000b
7	***Reserved.***	Yes	No	0

6.1.2.3 Timeout Control

Register 6-32. (TOCNTRL; PCI:45h) Timeout Control

Bit	Description	Read	Write	Value after Reset
2:0	**Maximum Retry Counter Control.** Controls the maximum number of times the PCI 6520 Retries a cycle before signaling a timeout. Timeout applies to Read/Write Retries and can be enabled to trigger SERR# on the primary or secondary port, depending on the SERR# events enabled. Maximum number of Retries to timeout: 000b = 2^{24} 001b = 2^{18} 010b = 2^{12} 011b = 2^{6} 111b = 2^{0}	Yes	Yes; Serial EEPROM	000b
3	***Reserved.***	Yes	No	0
5:4	**Primary Master Timeout Divider.** Provides additional options for the primary Master Timeout. In addition to its original setting in the Bridge Control register (BCNTRL[8]; PCI:3Eh), the Timeout Counter can optionally be divided by up to 256: 00b = Counter—Primary Master Timeout / 1 01b = Timeout Counter—Primary Master Timeout / 8 10b = Timeout Counter—Primary Master Timeout / 16 11b = Timeout Counter—Primary Master Timeout / 256 BCNTRL[8] can set the primary Master Timeout to 32K (default) or 1K Clock cycles.	Yes	Yes; Serial EEPROM	00b
7:6	**Secondary Master Timeout Divider.** Provides additional options for the secondary Master Timeout. In addition to its original setting in the Bridge Control register (BCNTRL[9]; PCI:3Eh), the Timeout Counter can optionally be divided by up to 256: 00b = Counter—Secondary Master Timeout / 1 01b = Timeout Counter—Secondary Master Timeout / 8 10b = Timeout Counter—Secondary Master Timeout / 16 11b = Timeout Counter—Secondary Master Timeout / 256 BCNTRL[9] can set the secondary Master Timeout to 32K (default) or 1K Clock cycles.	Yes	Yes; Serial EEPROM	00b

6.1.2.4 Miscellaneous Options

Register 6-33. (MSCOPT; PCI:46h) Miscellaneous Options

Bit	Description	Read	Write	Value after Reset
0	**Write Completion Wait for PERR#.** If set to 1, PCI 6520 waits for target PERR# status before completing a Delayed Write transaction to the initiator.	Yes	Yes; Serial EEPROM	0
1	**Read Completion Wait for PAR.** If set to 1, PCI 6520 waits for target PAR status before completing a Delayed Read transaction to the initiator.	Yes	Yes; Serial EEPROM	0
2	**DRT Out-of-Order Enable.** If set to 1, PCI 6520 may return Delayed Read transactions in a different order than requested. Otherwise, Delayed Read transactions are returned in the same order as requested.	Yes	Yes; Serial EEPROM	0
3	**Generate Parity Enable.** Values: 0 = Passes along the cycle PAR/PAR64, as stored in the internal buffers 1 = PCI 6520, as a master, generates PAR/PAR64 to cycles traveling across the bridge	Yes	Yes; Serial EEPROM	0

Register 6-33. (MSCOPT; PCI:46h) Miscellaneous Options (Continued)

Bit	Description	Read	Write	Value after Reset
6:4	**Address Step Control.** During Type 0 Configuration cycles, PCI 6520 drives the address for the number of clocks specified by these bits, before asserting FRAME#. Defaults to 001b in Conventional PCI mode. In PCI-X mode, these bits have no effect and address stepping is hardcoded to four clocks. Values: 000b = Concurrently asserts FRAME# and drives the address on the bus 001b = Asserts FRAME# one clock after driving the address on the bus … 111b = Asserts FRAME# seven clocks after driving the address on the bus	Yes	Yes; Serial EEPROM	001b
8:7	***Reserved.***	Yes	No	00b
9	**Prefetch Early Termination.** Values: 0 = Terminates prefetching at the Initial Prefetch Count if Flow Through is not achieved, and another Prefetching Read cycle is accepted by the PCI 6520 1 = Completes prefetching as programmed by the Prefetch Count registers, regardless of other outstanding prefetchable reads in the Transaction queue	Yes	Yes; Serial EEPROM	0
10	**Read Minimum Enable.** If set to 1, PCI 6520 only initiates Read cycles if there is sufficient space available in the FIFO as required by the Prefetch Count registers.	Yes	Yes; Serial EEPROM	0
15, 11	**Force 64-Bit Control.** If set and the target supports 64-bit transfers, 32-bit Prefetchable reads or 32-bit Posted Memory Write cycles on one side are converted to 64-bit cycles on completion at the target bus. If set to 00b, cycles are not converted. Values: 00b = Disable (default) 01b = Convert to 64-bit command on both ports 10b = Convert to 64-bit command on secondary port 11b = Convert to 64-bit command on primary port	Yes	Yes; Serial EEPROM	0
12	**Memory Write and Invalidate Control.** Values: 0 = Retries Memory Write and Invalidate commands if there is insufficient space for one cache line of data in the internal queues. 1 = Passes Memory Write and Invalidate commands if there are one or more cache lines of FIFO space available. If there is insufficient space, completes as a Memory Write cycle.	Yes	Yes; Serial EEPROM	0
13	**Primary Lock Enable.** If set to 1, PCI 6520 follows the LOCK protocol on primary interface; otherwise, LOCK is ignored.	Yes	Yes; Serial EEPROM	0
14	**Secondary Lock Enable.** If set to 1, PCI 6520 follows the LOCK protocol on secondary interface; otherwise, LOCK is ignored.	Yes	Yes; Serial EEPROM	0

6.1.2.5 Prefetch Control

Registers 48h to 4Eh are the Flow-Through Prefetch Control registers, which are used to fine-tune the PCI 6520 Memory Read prefetch behavior. (Refer to Section 18, "PCI Flow-Through Optimization," for further details regarding these registers.) These registers apply only if there are one or more ports operating in Conventional PCI mode.

Register 6-34. (PITLPCNT; PCI:48h) Primary Initial Prefetch Count

Bit	Description	Read	Write	Value after Reset
0	***Reserved.***	Yes	No	0
2:1	**PCI-X Primary initial Prefetch Count.** When the primary bus is in PCI-X mode, these bits control the initial Prefetch Count on the primary bus during reads to Prefetchable Memory space. Value defines the cache line multiples for the initial Prefetch Count. Prefetch as follows: 00b = 1 cache line (size defined by PCICLSR; PCI:0Ch) 01b = 2 cache lines 10b = 4 cache lines 11b = 8 cache lines The Primary PCI-X 16 Cache Line Prefetch bit (PINCPCNT[1]; PCI:4Ah) enables prefetch of 16 cache lines.	Yes	Yes; Serial EEPROM	00b
5:3	**PCI Primary initial Prefetch Count.** When the primary bus is in Conventional PCI mode, controls the Initial Prefetch Count on the primary bus during reads to Prefetchable Memory space. Prefetch as follows: 001b = 08h Dwords 010b = 10h Dwords 101b = 20h Dwords Other values are ***Reserved***.	Yes	Yes; Serial EEPROM	001b
7:6	***Reserved.***	Yes	No	00b

Register 6-35. (SITLPCNT; PCI:49h) Secondary Initial Prefetch Count

Bit	Description	Read	Write	Value after Reset
0	***Reserved.***	Yes	No	0
2:1	**PCI-X Secondary Initial Prefetch Count.** When the secondary bus is in PCI-X mode, these bits control the initial Prefetch Count on the secondary bus during reads initiated from the primary port. Value defines the cache line multiples for the initial Prefetch Count. Prefetch as follows: 00b = 1 cache line (size defined by PCICLSR; PCI:0Ch) 01b = 2 cache lines 10b = 4 cache lines 11b = 8 cache lines The Secondary PCI-X 16 Cache Line Prefetch bit (SINCPCNT[1]; PCI:4Bh) enables prefetch of 16 cache lines.	Yes	Yes; Serial EEPROM	00b
5:3	**PCI Secondary Initial Prefetch Count.** When the secondary bus is in Conventional PCI mode, these bits control the initial Prefetch Count on the secondary bus during reads initiated from the primary port. Prefetch as follows: 001b = 08h Dwords 010b = 10h Dwords 101b = 20h Dwords Other values are ***Reserved***.	Yes	Yes; Serial EEPROM	001b
6	**Primary Write Flush Enable.** Values: 0 = Downstream writes (any type) do not affect smart prefetch entries 1 = Flushes all active smart prefetch entries on downstream writes	Yes	Yes; Serial EEPROM	0
7	**Secondary Write Flush Enable.** Values: 0 = Upstream Memory writes do not affect smart prefetch entries 1 = Flushes the entry if the upstream write address hits 4-KB page of the smart prefetch entries	Yes	Yes; Serial EEPROM	0

Register 6-36. (PINCPCNT; PCI:4Ah) Primary Incremental Prefetch Count

Bit	Description	Read	Write	Value after Reset
0	***Reserved.***	Yes	No	0
1	**Primary PCI-X 16 Cache Line Prefetch.** Value: 1 = Enables primary PCI-X port to prefetch 16 cache lines. When set, the PCI-X Primary Initial Prefetch Count bits are ignored (PITLPCNT[2:1]; PCI:48h).	Yes	Yes; Serial EEPROM	0
5:2	**Primary Incremental Prefetch Count.** Applies only to primary port Conventional PCI mode operation. These bits control the incremental read prefetch count. When an entry's remaining prefetch Dword count falls below this value, the bridge prefetches additional primary Incremental Prefetch Count Dwords. Value is specified as a multiple of 4 x Dwords. The register value must not exceed half the value programmed in the Primary Maximum Prefetch Count register (PMAXPCNT; PCI:4Ch); otherwise, no incremental prefetch is performed. Prefetch as follows: 0000b = No incremental prefetch 0001b = 04h Dwords 0010b = 08h Dwords 0011b = 0Ch Dwords … 1111b = 3Ch Dwords	Yes	Yes; Serial EEPROM	0000b
7:6	***Reserved.***	Yes	No	00b

Register 6-37. (SINCPCNT; PCI:4Bh) Secondary Incremental Prefetch Count

Bit	Description	Read	Write	Value after Reset
0	***Reserved.***	Yes	No	0
1	**Secondary PCI-X 16 Cache Line Prefetch.** Value: 1 = Enables secondary PCI-X port to prefetch 16 cache lines. When set, the PCI-X Secondary Initial Prefetch Count bits are ignored (SITLPCNT[2:1]; PCI:49h).	Yes	Yes; Serial EEPROM	0
5:2	**Secondary Incremental Prefetch Count.** Applies only to secondary port Conventional PCI mode operation. These bits control the incremental read prefetch count. When an entry's remaining prefetch Dword count falls below this value, the bridge prefetches additional secondary incremental prefetch count Dwords. Value is specified as a multiple of 4 x Dwords. The register value must not exceed half the value programmed in the Secondary Maximum Prefetch Count register (SMAXPCNT; PCI:4Dh); otherwise, no incremental prefetch is performed. Prefetch as follows: 0000b = No incremental prefetch 0001b = 04h Dwords 0010b = 08h Dwords 0011b = 0Ch Dwords … 1111b = 3Ch Dwords	Yes	Yes; Serial EEPROM	0000b
7:6	***Reserved.***	Yes	No	00b

Register 6-38. (PMAXPCNT; PCI:4Ch) Primary Maximum Prefetch Count

Bit	Description	Read	Write	Value after Reset
5:0	**Primary Maximum Prefetch Count.** Applies only to PCI-to-PCI bridging. Limits the cumulative maximum count of prefetchable Dwords allocated to one entry on the primary bus when Flow Through for that entry is not achieved. Value should be an even number. Bit 0 is Read-Only and always 0. Value is specified in Dwords, *except* if 0 value is programmed, which sets the Primary Maximum Prefetch Count to its maximum value of 256 bytes. A PCI Read cycle causes a PCI request for the Maximum Count data.	Yes	Yes [5:1]; Serial EEPROM	20h
7:6	***Reserved.***	Yes	No	00b

Register 6-39. (SMAXPCNT; PCI:4Dh) Secondary Maximum Prefetch Count

Bit	Description	Read	Write	Value after Reset
5:0	**Secondary Maximum Prefetch Count.** Applies only to PCI-to-PCI bridging. Limits the cumulative maximum count of prefetchable Dwords allocated to one entry on the secondary bus when Flow Through for that entry is not achieved. Value should be an even number. Bit 0 is Read-Only and always 0. Value is specified in Dwords, *except* if 0 value is programmed, which sets the Secondary Maximum Prefetch Count to its maximum value of 256 bytes. A PCI Read cycle causes a PCI request for the Maximum Count data.	Yes	Yes [5:1]; Serial EEPROM	20h
7:6	***Reserved.***	Yes	No	00b

6.1.2.6 Secondary Flow-Through Control

Register 6-40. (SFTCR; PCI:4Eh) Secondary Flow-Through Control

Bit	Description	Read	Write	Value after Reset
2:0	**Secondary Posted Write Completion Wait Count.** Maximum number of clocks the PCI 6520 waits for Posted Write data from the initiator if delivering Write data in Flow-Through mode and the Internal Post Write queues are almost empty. If the count is exceeded without additional data from the initiator, the cycle to the target is terminated and later completed. Value: 000b = De-asserts P_IRDY# and waits seven clocks for data on the secondary bus, before terminating cycle All other values are ***Reserved.*** ***Note:*** *To specify other clock values, use software or the serial EEPROM to set these bits to any value between two and seven clocks.*	Yes	Yes; Serial EEPROM	000b
3	***Reserved.***	Yes	No	0
6:4	**Secondary Delayed Read Completion Wait Count.** Maximum number of clocks the PCI 6520 waits for Delayed Read data from the target, if returning Read data in Flow-Through mode and the Internal Delayed Read queue is almost full. If the count is exceeded without additional space in the queue, the cycle to target is terminated, and completed when initiator Retries the remainder of the cycle. Value: 000b = De-asserts P_IRDY# and waits seven clocks for additional space to be transferred to the secondary bus before terminating cycle All other values are ***Reserved.*** ***Note:*** *To specify other clock values, use software or the serial EEPROM to set these bits to any value between two and seven clocks.*	Yes	Yes; Serial EEPROM	000b
7	***Reserved.***	Yes	No	0

6.1.2.7 Buffer and Internal Arbiter Control

Register 6-41. (BUFCR; PCI:4Fh) Buffer Control

Bit	Description	Read	Write	Value after Reset
0	***Reserved.***	Yes	No	0
1	**Smart Prefetch Enable.** The amount of data prefetched is defined in the Maximum Prefetch Count registers (PMAXPCNT; PCI:4Ch and SMAXPCNT; PCI:4Dh). Values after a prefetch command: 0 = Remaining prefetched data is discarded upon completion of the current Read Command. 1 = Remaining prefetched data is ***not*** discarded, but remains available for the next Read Command with consecutive address. The prefetched data is only discarded upon a timeout. The timeout period can be programmed using the Smart Prefetch Timeout bits (BUFCR[6:5]; PCI:4Fh).	Yes	Yes	0
2	**Split FIFO Enable.** Buffer Split for individual data entries. Values: 0 = Entire FIFO shared among entries and the FIFO is undedicated 1 = FIFO divided into four equal parts, to be dedicated to each entry for Split Completions	Yes	Yes	0
4:3	***Reserved.***	Yes	No	00b
6:5	**Smart Prefetch Timeout.** Smart Prefetch Timeout affects only PCI-to-PCI, PCI-to-PCI-X, or PCI-X-to-PCI bridging applications. Prefetches ***cannot*** cross the 4-KB Address boundary. When Smart Prefetch is enabled, the prefetched data is only discarded upon a timeout. The timeout periods available are after: 00b = 32 PCI clocks 01b = 64 PCI clocks 10b = 128 PCI clocks 11b = 256 PCI clocks	Yes	Yes	11b
7	***Reserved.***	Yes	No	0

Register 6-42. (IACNTRL; PCI:50h) Internal Arbiter Control

Bit	Description	Read	Write	Value after Reset
0	**Low-Priority Group Fixed Arbitration.** If set to 1, the low-priority group uses fixed-priority arbitration; otherwise, rotating-priority arbitration is used.	Yes	Yes	0
1	**Low-Priority Group Arbitration Order.** Valid only when the low-priority arbitration group is set to a fixed arbitration scheme. Values: 0 = Priority decreases with bus master number. (*For example*, assuming Master 2 is set as the highest priority master, Master 3 retains higher priority than Master 4.) 1 = Priority increases with bus master number. (*For example*, assuming Master 2 is set as the highest priority master, Master 4 retains higher priority than Master 3. This order is relative to the master with the highest priority for this group, as specified in IACNTRL[7:4].	Yes	Yes	0
2	**High-Priority Group Fixed Arbitration.** If set to 1, the high-priority group uses the fixed-priority arbitration; otherwise, rotating-priority arbitration is used**.**	Yes	Yes	0
3	**High-Priority Group Arbitration Order.** Valid only when the high-priority arbitration group is set to a fixed arbitration scheme. Values: 0 = Priority decreases with bus master number. (*For example*, assuming Master 2 is set as the highest priority master, Master 3 retains higher priority than Master 4.) 1 = Priority increases with bus master number. (*For example,* assuming Master 2 is set as the highest priority master, Master 4 retains higher priority than Master 3.) This order is relative to the master with the highest priority for this group, as specified in IACNTRL[11:8].	Yes	Yes	0
7:4	**Highest Priority Master in Low-Priority Group.** Controls which master in the low-priority group retain the highest priority. Valid only if the group uses the fixed arbitration scheme. Values: 0000b = Master 0 retains highest priority 0001b = Master 1 retains highest priority … 1000b = PCI 6520 retains highest priority 1001b – 1111b = ***Reserved***	Yes	Yes	0000b
11:8	**Highest Priority Master in High-Priority Group.** Controls which master in the high-priority group retains the highest priority. Valid only if the group uses the fixed arbitration scheme. Values: 0000b = Master 0 retains highest priority 0001b = Master 1 retains highest priority … 1000b = PCI 6520 retains highest priority 1001b – 1111b = ***Reserved***	Yes	Yes	0000b
15:12	**Bus Grant Parking Control.** Controls bus grant behavior during idle. Value: 0h = Indicates the last master granted is parked All other values are ***Reserved.***	Yes	No	0h

6.1.2.8 Test and Serial EEPROM

Register 6-43. (TEST; PCI:52h) Test

Bit	Description	Read	Write	Value after Reset
0	**Serial EEPROM Autoload Control.** If set to 1, disables serial EEPROM autoload.	Yes	Yes	0
1	**Fast Serial EEPROM Autoload.** If set to 1, speeds up serial EEPROM autoload.	Yes	Yes	0
2	**Serial EEPROM Autoload Status.** Serial EEPROM autoload status is set to 1 during autoload.	Yes	No	Status of Serial EEPROM Autoload
3	**S_PLL_TEST.** When P_CLKOE input pin is set to 1 and this bit is set to 1, S_CLKO4 (derived from S_CLKIN) is divided by 4.	Yes	Yes	0
4	**DEV64#.** Reflects DEV64# pin status.	Yes	No	DEV64#
5	**S_CFN#.** Reflects S_CFN# pin status.	Yes	No	S_CFN#
7:6	***Reserved.*** Returns 0 when read.	Yes	No	00b

Register 6-44. (EEPCNTRL; PCI:54h) Serial EEPROM Control

Bit	Description	Read	Write	Value after Reset
0	**Start.** Starts serial EEPROM Read or Write cycle. Bit is cleared when serial EEPROM load completes.	Yes	Yes	0
1	**Serial EEPROM Command.** Controls commands sent to the serial EEPROM. Values: 0 = Read 1 = Write	Yes	Yes	0
2	**Serial EEPROM Error.** Set to 1 if serial EEPROM ACK was not received during serial EEPROM cycle.	Yes	No	—
3	**Serial EEPROM Autoload Successful.** Set to 1 if serial EEPROM autoload successfully occurred after reset, with appropriate Configuration registers loaded with the values programmed in the serial EEPROM. If 0, the serial EEPROM autoload was unsuccessful or disabled.	Yes	No	—
5:4	***Reserved.*** Returns 00b when read.	Yes	No	00b
7:6	**Serial EEPROM Clock Rate.** Controls the serial EEPROM clock frequency. The serial EEPROM clock is derived from the primary PCI clock. Values: 00b = PCLK / 2048 01b = PCLK / 1024 10b = PCLK / 256 11b = PCLK / 32	Yes	Yes	01b

Register 6-45. (EEPADDR; PCI:55h) Serial EEPROM Address

Bit	Description	Read	Write	Value after Reset
0	***Reserved.***	Yes	No	—
7:1	**Serial EEPROM Address.** Word address for the serial EEPROM cycle.	Yes	Yes	—

Register 6-46. (EEPDATA; PCI:56h) Serial EEPROM Data

Bit	Description	Read	Write	Value after Reset
15:0	**Serial EEPROM Data.** Contains data to be written to the serial EEPROM. During reads, contains data received from the serial EEPROM after a Read cycle completes.	Yes	Yes	—

6.1.2.9 Timer

Register 6-47. (TMRCNTRL; PCI:61h) Timer Control

Bit	Description	Read	Write	Value after Reset
0	**Timer Enable.** Set to start measurement of the approximate bus frequency on the primary or secondary interface. By default, bit is 0, and must be set to 1 to start the measurement. When set to 0 and then to 1, PCI 6520 starts counting up until it reaches the set count period. During this counting period, PCI 6520 Timer Counter (TMRCNT; PCI:62h) counts the number of Timer Counter clocks.	Yes	Yes	0
2:1	**Timer Counter Clock Source Select.** Values: 00b = Primary PCI clock (P_CLKIN) 01b = Secondary PCI clock (S_CLKIN) 10b, 11b = ***Reserved***	Yes	Yes	00b
3	**Timer Stop.** Timer stopped status bit. When the measurement is finished, this bit is set to 0, and then to 1. When starting a new measurement, this bit automatically restores to 0. Values: 0 = Timer running 1 = Timer stopped	Yes	No	0
5:4	**Count Period.** Values: 00b = 16 Reference clock high states 01b = 32 Reference clock high states 10b = 64 Reference clock high states 11b = 128 Reference clock high states	Yes	Yes	00b
7:6	***Reserved.***	Yes	No	00b

Register 6-48. (TMRCNT; PCI:62h) Timer Counter

Bit	Description	Read	Write	Value after Reset
15:0	**Timer Counter.** Automatically stops upon the count period setting in the Timer Control register (TMRCNTRL[7:4]; PCI:61h). This counter can be enabled by setting the Timer Enable bit (TMRCNTRL[0]; PCI:61h) first to 0, and then to 1.	Yes	No	0h

6.1.2.10 Primary System Error Event

Register 6-49. (PSERRED; PCI:64h) P_SERR# Event Disable

Bit	Description	Read	Write	Value after Reset
0	***Reserved.***	Yes	No	0
1	**Posted Write Parity Error.** Controls PCI 6520 ability to assert P_SERR# when a Data Parity error is detected on the target bus during a Posted Write transaction. P_SERR# is asserted if this event occurs when bit is 0 and Command register P_SERR# Enable bit is set (PCICR[8]=1; PCI:04h).	Yes	Yes	0
2	**Posted Memory Write Non-Delivery.** Controls PCI 6520 ability to assert P_SERR# when it is unable to deliver Posted Write data after 2^{24} attempts [or programmed Maximum Retry count (TOCNTRL[2:0]; PCI:45h)]. P_SERR# is asserted if this event occurs when bit is 0 and Command register P_SERR# Enable bit is set (PCICR[8]=1; PCI:04h).	Yes	Yes	0
3	**Target Abort during Posted Write.** Controls PCI 6520 ability to assert P_SERR# when it receives a Target Abort while attempting to deliver Posted Write data. P_SERR# is asserted if this event occurs when bit is 0 and Command register P_SERR# Enable bit is set (PCICR[8]=1; PCI:04h).	Yes	Yes	0
4	**Master Abort on Posted Write.** Controls PCI 6520 ability to assert P_SERR# when it receives a Master Abort while attempting to deliver Posted Write data. P_SERR# is asserted if this event occurs when bit is 0 and Command register P_SERR# Enable bit is set (PCICR[8]=1; PCI:04h).	Yes	Yes	0
5	**Delayed Configuration or I/O Write Non-Delivery.** Controls PCI 6520 ability to assert P_SERR# when it is unable to deliver Delayed Write data after 2^{24} attempts [or programmed Maximum Retry count (TOCNTRL[2:0]; PCI:45h)]. P_SERR# is asserted if this event occurs when bit is 0 and Command register P_SERR# Enable bit is set (PCICR[8]=1; PCI:04h).	Yes	Yes	0
6	**Delayed Read-No Data from Target.** Controls PCI 6520 ability to assert P_SERR# when it is unable to transfer Read data from the target after 2^{24} attempts [or programmed Maximum Retry count (TOCNTRL[2:0]; PCI:45h)]. P_SERR# is asserted if this event occurs when bit is 0 and Command register P_SERR# Enable bit is set (PCICR[8]=1; PCI:04h).	Yes	Yes	0
7	***Reserved.*** Returns 0 when read.	Yes	No	0

6.1.2.11 GPIO[3:0]

Register 6-50. (GPIOOD[3:0]; PCI:65h) GPIO[3:0] Output Data

Bit	Description	Read	Write	Value after Reset
3:0	**GPIO[3:0] Output Data Write 1 to Clear.** Writing 1 to these bits drives the corresponding signal low on the GPIO[3:0] bus, if the signal is programmed as an output. Writing 0 has no effect. Read returns last written value.	Yes	Yes/Clr	0h
7:4	**GPIO[3:0] Output Data Write 1 to Set.** Writing 1 to these bits drives the corresponding signal high on the GPIO[3:0] bus, if the signal is programmed as an output. Writing 0 has no effect. Read returns last written value.	Yes	Yes/Set High	0h

Register 6-51. (GPIOOE[3:0]; PCI:66h) GPIO[3:0] Output Enable

Bit	Description	Read	Write	Value after Reset
3:0	**GPIO[3:0] Output Enable Write 1 to Clear.** Writing 1 to these bits configures the corresponding signal on the GPIO[3:0] bus as an input. Writing 0 has no effect. Read returns last written value.	Yes	Yes/Clr	0h
7:4	**GPIO[3:0] Output Enable Write 1 to Set.** Writing 1 to these bits configures the corresponding signal on the GPIO[3:0] bus as an output. Writing 0 has no effect. Read returns last written value.	Yes	Yes/Set High	0h

Register 6-52. (GPIOID[3:0]; PCI:67h) GPIO[3:0] Input Data

Bit	Description	Read	Write	Value after Reset
3:0	***Reserved.***	Yes	No	0h
7:4	**GPIO[3:0] Input Data.** Reads the GPIO[3:0] pin state. The state is updated on the primary PCI Clock cycle, following a change in GPIO[3:0] state.	Yes	No	—

6.1.2.12 Clock Control

Register 6-53. (CLKCNTRL; PCI:68h) Clock Control

Bit	Description	Read	Write	Value after Reset
1:0	**Clock 0 Disable**. If either bit is 0, S_CLKO0 is enabled. When both bits are 1, S_CLKO0 is disabled. Defaults to 00b if MSK_IN=1.	Yes	Yes	00b
3:2	**Clock 1 Disable.** If either bit is 0, S_CLKO1 is enabled. When both bits are 1, S_CLKO1 is disabled.	Yes	Yes	00b
5:4	**Clock 2 Disable.** If either bit is 0, S_CLKO2 is enabled. When both bits are 1, S_CLKO2 is disabled.	Yes	Yes	00b
7:6	**Clock 3 Disable.** If either bit is 0, S_CLKO3 is enabled. When both bits are 1, S_CLKO3 is disabled.	Yes	Yes	00b
8	**Clock 4 Disable.** If 0, S_CLKO4 is enabled. When 1, S_CLKO4 is disabled.	Yes	Yes	0
15:9	***Reserved.***	Yes	No	0h

6.1.2.13 Primary System Error Status

Register 6-54. (PSERRSR; PCI:6Ah) P_SERR# Status

Bit	Description	Read	Write	Value after Reset
0	**Address Parity Error.** P_SERR# is asserted because an Address Parity error occurred on either side of the bridge.	Yes	Yes/Clr	0
1	**Posted Write Data Parity Error.** P_SERR# is asserted because a Posted Write Data Parity error occurred on the target bus.	Yes	Yes/Clr	0
2	**Posted Write Non-Delivery.** P_SERR# is asserted because PCI 6520 was unable to deliver Posted Write data to the target before the Timeout Counter expired.	Yes	Yes/Clr	0
3	**Target Abort during Posted Write.** P_SERR# is asserted because PCI 6520 received a Target Abort when delivering Posted Write data.	Yes	Yes/Clr	0
4	**Master Abort during Posted Write.** P_SERR# is asserted because PCI 6520 received a Master Abort when delivering Posted Write data.	Yes	Yes/Clr	0
5	**Delayed Write Non-Delivery.** P_SERR# is asserted because PCI 6520 was unable to deliver Delayed Write data before the Timeout Counter expired.	Yes	Yes/Clr	0
6	**Delayed Read Failed.** P_SERR# is asserted because PCI 6520 was unable to read data from the target before the Timeout Counter expired.	Yes	Yes/Clr	0
7	**Delayed Transaction Master Timeout.** P_SERR# is asserted because a master did not repeat a Read or Write transaction before the initiator bus Master Timeout Counter expired.	Yes	Yes/Clr	0

6.1.2.14 Clock Run

Register 6-55. (CLKRUN; PCI:6Bh) Clock Run

Bit	Description	Read	Write	Value after Reset
0	**Secondary Clock Stop Status.** Values: 0 = Secondary clock not stopped 1 = Secondary clock stopped	Yes	Yes	0
1	**S_CLKRUN# Enable.** Controls the S_CLKRUN# pin. Values: 0 = Disables S_CLKRUN# pin 1 = Enables S_CLKRUN# pin	Yes	Yes	0
2	**Primary Clock Stop.** Values: 0 = Allows primary clock to stop, if secondary clock is stopped 1 = Keeps primary clock running	Yes	Yes	0
3	**P_CLKRUN# Enable.** Controls the P_CLKRUN# pin. Values: 0 = Disables P_CLKRUN# pin 1 = Enables P_CLKRUN# pin	Yes	Yes	0
4	**Clkrun Mode.** Values: 0 = Stops the secondary clock only on request from the primary bus 1 = Stops the secondary clock when the secondary bus is idle and there are no requests from the primary bus	Yes	Yes	0
7:5	***Reserved.***	Yes	No	000b

6.1.2.15 Private Memory

Private Memory can be enabled by way of the Chip Control register (CCNTRL[2]; PCI:40h) or by using the PRV_DEV input pin.

Register 6-56. (PVTMBAR; PCI:6Ch) Private Memory Base

Bit	Description	Read	Write	Value after Reset
15:0	**Private Memory Base.** Defines the Private Memory Address range Base address. The upper 12 bits corresponding to Address bits [31:20] are writable and reset to 0h. The lower 20 Address bits [19:0] are assumed to be 0h. The lower four bits are Read-Only and set to 0001b.	Yes	Yes [15:4]	1h

Register 6-57. (PVTMLMT; PCI:6Eh) Private Memory Limit

Bit	Description	Read	Write	Value after Reset
15:0	**Private Memory Limit.** Defines the Private Memory Address range Upper Limit address. The upper 12 bits corresponding to Address bits [31:20] are writable and reset to 0h. The lower 20 Address bits [19:0] are assumed to be F_FFFFh. The lower four bits are Read-Only and set to 0h.	Yes	Yes [15:4]	0h

Register 6-58. (PVTMBARU32; PCI:70h) Private Memory Base Upper 32 Bits

Bit	Description	Read	Write	Value after Reset
31:0	**Private Memory Base Upper 32 Bits**. Defines the upper 32-bit (bits [63:32]) Memory Base address of the Private Memory Address range. ***Note:*** *Private Memory Base default value is higher than the Private Memory Limit, and ensures that the Private Memory space is disabled by default.*	Yes	Yes	1h

Register 6-59. (PVTMLMTU32; PCI:74h) Private Memory Limit Upper 32 Bits

Bit	Description	Read	Write	Value after Reset
31:0	**Private Memory Limit Upper 32 Bits.** Defines the upper 32-bit (bits [63:32]) Memory Limit address of the Private Memory Address range.	Yes	Yes	0h

6.1.2.16 Read-Only Register Control

Register 6-60. (RRC; PCI:9Ch) Read-Only Register Control

Bit	Description	Read	Write	Value after Reset
0	***Reserved.***	Yes	No	0
1	**Primary Port 64-Bit Extension Signals Park.** Value: 1 = Drives primary port PCI 64-bit extension signals P_AD[63:32], P_CBE[7:4]#, and P_PAR64 to 0	Yes	Yes	0
2	**Secondary Port 64-Bit Extension Signals Park.** Value: 1 = Drives secondary port PCI 64-bit extension signals S_AD[63:32], S_CBE[7:4]#, and S_PAR64 to 0	Yes	Yes	0
6:3	***Reserved. Must be 0h.***	Yes	No	0h
7	**Read-Only Registers Write Enable.** Setting this bit to 1 enables writes to specific bits within these normally Read-Only registers (refer to the listed registers for further details): • Vendor and Device IDs (PCIIDR; PCI:00h) • PCI Class Code (PCICCR; PCI:09h – 0Bh) • PCI Header Type (PCIHTR; PCI:0Eh) • Power Management Capabilities (PMC; PCI:DEh) • Power Management Control/Status (PMCSR; PCI:E0h) • Power Management Data (PMCDATA; PCI:E3h) Bit must be cleared after the values are modified in these Read-Only registers.	Yes	Yes	0

6.1.2.17 GPIO[7:4]

Register 6-61. (GPIOOD[7:4]; PCI:9Dh) GPIO[7:4] Output Data

Bit	Description	Read	Write	Value after Reset
3:0	**GPIO[7:4] Output Data Write 1 to Clear.** Writing 1 to these bits drives the corresponding signal low on the GPIO[7:4] bus, if the signal is programmed as an output. Writing 0 has no effect. Read returns last written value.	Yes	Yes/Clr	0h
7:4	**GPIO[7:4] Output Data Write 1 to Set.** Writing 1 to these bits drives the corresponding signal high on the GPIO[7:4] bus, if the signal is programmed as an output. Writing 0 has no effect. Read returns last written value.	Yes	Yes/Set High	0h

Register 6-62. (GPIOOE[7:4]; PCI:9Eh) GPIO[7:4] Output Enable

Bit	Description	Read	Write	Value after Reset
3:0	**GPIO[7:4] Output Enable Write 1 to Clear.** Writing 1 to these bits configures the corresponding signal on the GPIO[7:4] bus as an input. Writing 0 has no effect. Read returns last written value.	Yes	Yes/Clr	0h
7:4	**GPIO[7:4] Output Enable Write 1 to Set.** Writing 1 to these bits configures the corresponding signal on the GPIO[7:4] bus as an output. Writing 0 has no effect. Read returns last written value.	Yes	Yes/Set High	0h

Register 6-63. (GPIOID[7:4]; PCI:9Fh) GPIO[7:4] Input Data

Bit	Description	Read	Write	Value after Reset
3:0	***Reserved.***	Yes	No	0h
7:4	**GPIO[7:4] Input Data.** Reads the GPIO[7:4] pin state. The state is updated on the primary PCI Clock cycle following a change in GPIO[7:4] state.	Yes	No	—

6.1.2.18 Extended and Smart Prefetch

The extended registers are accessed by way of the Extended Register Index and Data registers (EXTRIDX; PCI:D3h and EXTRDATA; PCI:D4h, respectively).

There are six 32-bit Upstream Smart Prefetch BARs (EXT:10h to 15h) and four 32-bit Upstream Smart Prefetch Descriptor registers (EXT:16h to 19h). (Refer to Table 6-2.)

Smart Prefetch is enabled by way of the Buffer Control register (BUFCR[1]=1; PCI:4Fh). The secondary bus Upstream Smart Prefetch Memory Space is divided into four regions. Regions 1, 2, and 3 (smart prefetchable) are actively decoded. Region 4 (non-smart prefetchable) is the region exclusive of regions 1, 2, and 3 that are decoded by the PCI 6520, where the cycles are passed upstream to the host. The three actively decoded Smart Prefetch regions require a 64-bit BAR type register for address allocation. The Smart Prefetch Region window size is defined in each region's descriptor register. After one of the Smart Prefetchable regions (1, 2, or 3) is enabled, Region 4 is enabled with the default setting, even though bit 31 of the Region 4 Descriptor register is set to 0. (Refer to Figure 6-1.)

The initiated Smart Prefetch cycle does not prefetch data across the 4-KB boundary. All regions monitor the upstream Write cycles. If a Write cycle occurs in one of the smart prefetched 4-KB data pages, the Write cycle causes that entry to be flushed.

The Smart Prefetch BAR registers for Regions 1, 2, and 3 are located in Extended Configuration register space. The Extended registers can be accessed by writing the 8-bit offset to the Extended Register Index register (EXTRIDX; PCI:D3h), then read or write the 32-bit data from or to the Extended Register Data register (EXTRDATA; PCI:D4h). Each Smart Prefetch region has one 8-bit descriptor register to define that region's configuration. (Refer to Table 6-2 on page 6-41 and the registers that follow.)

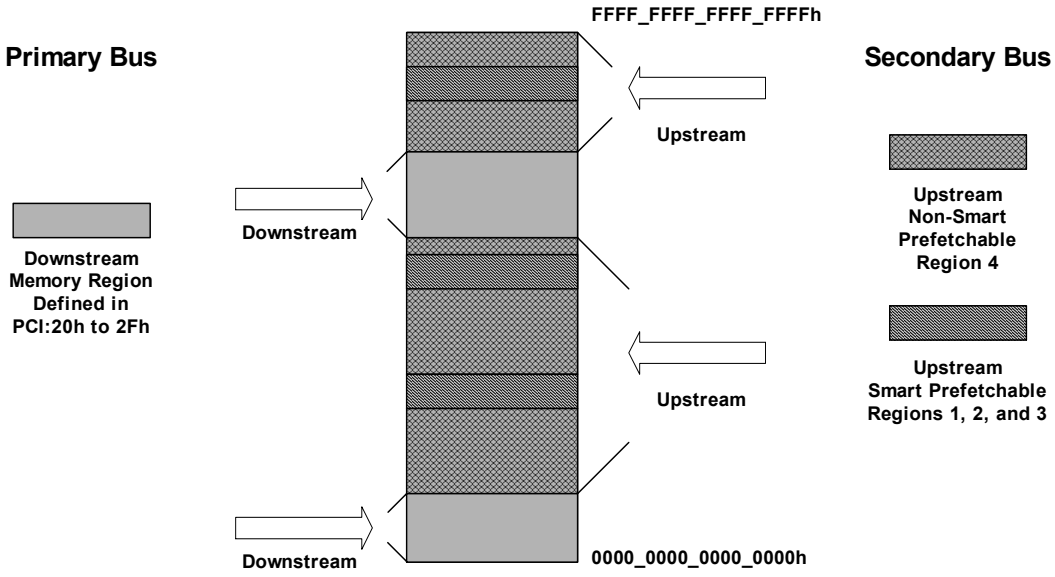


Figure 6-1. Sample Memory Map of Smart Prefetch Upstream Memory, Regions 1 through 4

Register 6-64. (EXTRIDX; PCI:D3h) Extended Register Index

Bit	Description	Read	Write	Value after Reset
7:0	**Extended Index Address.** Index address for Extended registers.	Yes	Yes	—

Register 6-65. (EXTRDATA; PCI:D4h) Extended Register Data

Bit	Description	Read	Write	Value after Reset
31:0	**Extended Register Data.** Configuration Write causes the data presented at this port to be written into the register addressed by the Extended Register Index (EXTRIDX; PCI:D3h). Configuration Read causes the data from the register addressed by the Extended Register Index to be placed into and read from EXTRDATA.	Yes	Yes	—

Table 6-2. Extended Register Map—Offset from Extended Register Index

Extended Register Index	31 24 23 16 15 8 7 0	Writable	Serial EEPROM Writable
10h	Region 1 Upstream Lower 32-Bit Smart Prefetch BAR	Yes	No
11h	Region 1 Upstream Upper 32-Bit Smart Prefetch BAR	Yes	No
12h	Region 2 Upstream Lower 32-Bit Smart Prefetch BAR	Yes	No
13h	Region 2 Upstream Upper 32-Bit Smart Prefetch BAR	Yes	No
14h	Region 3 Upstream Lower 32-Bit Smart Prefetch BAR	Yes	No
15h	Region 3 Upstream Upper 32-Bit Smart Prefetch BAR	Yes	No
16h	Region 1 Upstream Smart Prefetch BAR Descriptor	Yes	No
17h	Region 2 Upstream Smart Prefetch BAR Descriptor	Yes	No
18h	Region 3 Upstream Smart Prefetch BAR Descriptor	Yes	No
19h	Region 4 Upstream Smart Prefetch BAR Descriptor	Yes	No

Note: *Refer to the individual register descriptions to determine which bits are writable.*

Register 6-66. (SPUL32BAR1; EXT:10h) Region 1 Upstream Lower 32-Bit Smart Prefetch BAR

Bit	Description	Read	Write	Value after Reset
19:0	***Reserved.*** 1 MB Memory Address boundary resolution. ***Must be 0.***	Yes	Yes	0h
31:20	**Region 1 Upstream Lower 32-Bit Address for AD[31:0] of Base Address.**	Yes	Yes	0h

Register 6-67. (SPUU32BAR1; EXT:11h) Region 1 Upstream Lower 32-Bit Smart Prefetch BAR

Bit	Description	Read	Write	Value after Reset
31:0	**Region 1 Upstream Upper 32-Bit Address for AD[63:32] of Base Address.**	Yes	Yes	0h

Register 6-68. (SPUL32BAR2; EXT:12h) Region 2 Upstream Lower 32-Bit Smart Prefetch BAR

Bit	Description	Read	Write	Value after Reset
19:0	***Reserved.*** 1 MB Memory Address boundary resolution. ***Must be 0.***	Yes	Yes	0h
31:20	**Region 2 Upstream Lower 32-Bit Address for AD[31:0] of Base Address.**	Yes	Yes	0h

Register 6-69. (SPUU32BAR2; EXT:13h) Region 2 Upstream Lower 32-Bit Smart Prefetch BAR

Bit	Description	Read	Write	Value after Reset
31:0	**Region 2 Upstream Upper 32-Bit Address for AD[63:32] of Base Address.**	Yes	Yes	0h

Register 6-70. (SPUL32BAR3; EXT:14h) Region 3 Upstream Lower 32-Bit Smart Prefetch BAR

Bit	Description	Read	Write	Value after Reset
19:0	***Reserved.*** 1 MB Memory Address boundary resolution. ***Must be 0.***	Yes	Yes	0h
31:20	**Region 3 Upstream Lower 32-Bit Address for AD[31:0] of Base Address.**	Yes	Yes	0h

Register 6-71. (SPUU32BAR3; EXT:15h) Region 3 Upstream Lower 32-Bit Smart Prefetch BAR

Bit	Description	Read	Write	Value after Reset
31:0	**Region 1 Upstream Upper 32-Bit Address for AD[63:32] of Base Address.**	Yes	Yes	0h

Register 6-72. (SPUBARD*x*; EXT:16h–19h) Regions 1–4 Upstream Smart Prefetch BAR Descriptors

Bit	Description	Read	Write	Value after Reset
1:0	**Smart Prefetch Discard Timer.** Counting begins at end of Initiator access. Values: 00b = 32 clocks 01b = 64 clocks 10b = 128 clocks 11b = 256 clocks	Yes	Yes	00b
4:2	***Reserved. Must be set to 0.***	Yes	No	000b
5	**MEMR Command Flow-Through Enable.** Values: 0 = Disables MEMR Flow Through, prefetches only up to initial count (default) 1 = Enables MEMR Flow Through, prefetches up to the initial count, then continues to read until initiator disconnects	Yes	Yes	0
8:6	**MEMR Command Prefetch Count Multiplier.** Values: 000b = Prefetches until end of boundary of one Cache Line block (default) 001b = Prefetches until end of boundary of two Cache Line blocks 010b = Prefetches until end of boundary of four Cache Line blocks 011b = Prefetches until end of boundary of 8 Cache Line blocks 100b = Prefetches until end of boundary of 16 Cache Line blocks 101b, 110b, and 111b = ***Reserved***	Yes	Yes	000b
9	**MEMRL Command Flow-Through Enable.** Values: 0 = Disables MEMRL Flow Through, prefetches only up to initial count (default) 1 = Enables MEMRL Flow Through, prefetches up to the initial count, then continues to read until initiator disconnects	Yes	Yes	0

Register 6-72. (SPUBARD*x*; EXT:16h–19h) Regions 1–4 Upstream Smart Prefetch BAR Descriptors (Continued)

Bit	Description	Read	Write	Value after Reset
12:10	**MEMRL Command Prefetch Count Multiplier.** Values: 000b = Prefetches until end of boundary of one Cache Line block (default) 001b = Prefetches until end of boundary of two Cache Line blocks 010b = Prefetches until end of boundary of four Cache Line blocks 011b = Prefetches until end of boundary of 8 Cache Line blocks 100b = Prefetches until end of boundary of 16 Cache Line blocks 101b, 110b, and 111b = ***Reserved***	Yes	Yes	000b
13	**MEMRM Command Flow-Through Enable.** Values: 0 = Disables MEMRM Flow Through, prefetches only up to initial count 1 = Enables MEMRM Flow Through, prefetches up to the initial count, then continues to read until initiator disconnects (default)	Yes	Yes	1
16:14	**MEMRM Command Prefetch Count Multiplier.** Values: 000b = Prefetches until end of boundary of one Cache Line block 001b = Prefetches until end of boundary of two Cache Line blocks (default) 010b = Prefetches until end of boundary of four Cache Line blocks 011b = Prefetches until end of boundary of 8 Cache Line blocks 100b = Prefetches until end of boundary of 16 Cache Line blocks 101b, 110b, and 111b = ***Reserved***	Yes	Yes	000b
17	**Smart Prefetch Enable.** Values: 0 = Releases entry after initiator completes its cycle (default, no Smart Prefetch function). 1 = Retains entry if data remains after initiator completes its cycle. Data is released if one of the following conditions is met: • Discard Timer expires (SPUBARD*x*[1:0]) • Upon upstream write, if the Secondary Initial Prefetch Count Primary Write Flush Enable bit is set (SITLPCNT[6]=1; PCI:49h) • Upon upstream write within 4-KB page of the Smart Prefetch region if the Secondary Initial Prefetch Count Secondary Write Flush Enable bit is set (SITLPCNT[7]=1; PCI:49h)	Yes	Yes	0
19:18	**Smart Prefetch Region Cache Line Size.** Values: 00b = 8 Dwords (default) 01b = 16 Dwords 10b = 32 Dwords 11b = 64 Dwords	Yes	Yes	00b

Register 6-72. (SPUBARD*x*; EXT:16h–19h) Regions 1–4 Upstream Smart Prefetch BAR Descriptors (Continued)

Bit	Description	Read	Write	Value after Reset
20	**Smart Prefetch Region Cache Line Size Select.** Values: 0 = Uses Cache Line Size register value (PCICLSR; PCI:0Ch) (default) 1 = Uses Cache Line Size defined in SPUBARD*x*[19:18]	Yes	Yes	0
23:21	**Region 1, 2, and 3 Smart Prefetch BAR Window Size.** Values: 000b = 1 MB window (default) 001b = 2 MB window 010b = 4 MB window 011b = 8 MB window 100b = 64 MB window 101b = 128 MB window 110b = 256 MB window 111b = 512 MB window	Yes	Yes	000b
25:24	**Prefetch Disconnect Policy.** Values: 00b = Stops prefetching on the earliest of an initiator or target termination (default) 01b = Ignores initiator termination, then prefetches until requested count is fulfilled, unless target prematurely disconnects 10b = ***Reserved*** 11b = Ignores initiator termination, then prefetches until requested count is fulfilled	Yes	Yes	00b
30:26	***Reserved. Must be set to 0.***	Yes	No	0h
31	**Smart Prefetch BAR Region Enable.** Values: 0 = Disables Smart Prefetch BAR Regions 1, 2, and 3 (default) 1 = Enables Smart Prefetch BAR Regions 1, 2, and 3 ***Note:*** *Region 4 is automatically enabled (although value is 0) when Region 1, 2, and/or 3 is enabled.*	Yes	Yes	0

Note: *The "x" in SPUBARDx represents the Region numbers, 1 through 4 (that is, SPUBARD1 is the register for Region 1, SPUBARD2 is the register for Region 2, and so forth).*

6.1.2.19 Power Management Capability

Specific bits in the PMC; PCI:DEh, PMCDATA; PCI:E3h, and PMCSR; PCI:E0h Power Management registers are normally Read-Only. However, their default values can be changed by firmware or software by setting the Read-Only Registers Write Enable bit (RRC[7]=1; PCI:9Ch). After modifying these registers, the Write Enable bit must be cleared to preserve the Read-Only nature of these registers. It should be noted that the RRC[7] state does ***not*** affect Write accesses to PMCSR[15, 8].

Register 6-73. (PMCAPID; PCI:DCh) Power Management Capability ID

Bit	Description	Read	Write	Value after Reset
7:0	**Power Management Capability ID.** PCI-SIG-issued Capability ID for Power Management is 1h.	Yes	No	1h

Register 6-74. (PMNEXT; PCI:DDh) Power Management Next Capability Pointer

Bit	Description	Read	Write	Value after Reset
7:0	**Next_Cap Pointer.** Provides an offset into PCI Configuration space for the VPD capability location in the New Capabilities Linked List.	Yes	No	E8h

Register 6-75. (PMC; PCI:DEh) Power Management Capabilities

Bit	Description	Read	Write	Value after Reset
2:0	**Version.** Set to 001b, indicates that this function complies with *PCI Power Mgmt. r1.1*.	Yes	Only if RRC[7]=1; Serial EEPROM	001b
3	**PME Clock.** Set to 0, because PCI 6520 does not support PME# signaling.	Yes	Only if RRC[7]=1; Serial EEPROM	0
4	**Auxiliary Power Source.** Set to 0, because PCI 6520 does not support PME# signaling.	Yes	Only if RRC[7]=1; Serial EEPROM	0
5	**Device-Specific Initialization (DSI).** Returns 0, indicating PCI 6520 does not require special initialization.	Yes	Only if RRC[7]=1; Serial EEPROM	0
8:6	***Reserved.***	Yes	No	000b
9	**D_1 Support.** Returns 1, indicating that PCI 6520 supports the D_1 device power state.	Yes	Only if RRC[7]=1; Serial EEPROM	1
10	**D_2 Support.** Returns 1, indicating that PCI 6520 supports the D_2 device power state.	Yes	Only if RRC[7]=1; Serial EEPROM	1
15:11	**PME Support.** Indicates the power states in which the function may assert PME#. Value of 0 for any bit indicates that the function is not capable of asserting PME# while in that power state. Values: XXXX1b = PME# can be asserted from D_0 XXX1Xb = PME# can be asserted from D_1 XX1XXb = PME# can be asserted from D_2 X1XXXb = PME# can be asserted from D_{3hot} 1XXXXb = PME# can be asserted from D_{3cold}	Yes	Only if RRC[7]=1; Serial EEPROM	01111b

Register 6-76. (PMCSR; PCI:E0h) Power Management Control/Status

Bit	Description	Read	Write	Value after Reset
1:0	**Power State.** Used to determine the current power state of a function and to set the function into a new power state. Values: 00b = D_0 (default) 01b = D_1; valid only if PMC[9]=1; PCI:DEh 10b = D_2; valid only if PMC[10]=1; PCI:DEh 11b = D_{3hot}; if BPCC_EN=1, S_CLKO[4:0] are stopped	Yes	Yes; Serial EEPROM	00b
7:2	***Reserved.***	Yes	No	0h
8	**PME Enable.** Set to 0, because PCI 6520 does not support PME# signaling.	Yes	Yes; Serial EEPROM	0
12:9	**Data Select.** Returns 0h, indicating PCI 6520 does not return dynamic data.	Yes	Only if RRC[7]=1; Serial EEPROM	0h
14:13	**Data Scale.** Returns 00b when read. PCI 6520 does not return dynamic data.	Yes	No	00b
15	**PME Status.** Set to 0, because PCI 6520 does not support PME# signaling.	Yes	Yes; Serial EEPROM	0

Register 6-77. (PMCSR_BSE; PCI:E2h) PMCSR Bridge Supports Extensions

Bit	Description	Read	Write	Value after Reset
5:0	***Reserved.***	Yes	No	0h
6	**B_2/B_3 Support for D_{3hot}.** Reflects BPCC_EN input pin state. Value of 1 indicates that when PCI 6520 is programmed to D_{3hot} state, the secondary bus clock is stopped.	Yes	No	—
7	**Bus Power Control Enable.** Reflects BPCC_EN input pin state. Value of 1 indicates that the secondary bus Power Management state follows that of PCI 6520, with one exception—D_{3hot}.	Yes	No	—

Register 6-78. (PMCDATA; PCI:E3h) Power Management Data

Bit	Description	Read	Write	Value after Reset
7:0	**Power Management Data.** Serial EEPROM or Read-Only Register (ROR) Write controlled loadable, but is Read-Only during normal operation.	Yes	Only if RRC[7]=1; Serial EEPROM	0h

6.1.2.20 VPD Capability

Register 6-79. (PVPDID; PCI:E8h) Vital Product Data Capability ID

Bit	Description	Read	Write	Value after Reset
7:0	**Vital Product Data Capability ID.** PCI-SIG-issued Capability ID for VPD is 03h.	Yes	No	03h

Register 6-80. (PVPD_NEXT; PCI:E9h) Vital Product Data Next Capability Pointer

Bit	Description	Read	Write	Value after Reset
7:0	**Next_Cap Pointer**. Provides offset into PCI Configuration space for the PCI-X capability location in the New Capabilities Linked List.	Yes	No	F0h

Register 6-81. (PVPDAD; PCI:EAh) Vital Product Data Address

Bit	Description	Read	Write	Value after Reset
1:0	***Reserved.***	Yes	No	00b
7:2	**VPD Address.** Offset into the serial EEPROM to location where data is written and read. PCI 6520 accesses the serial EEPROM at address PVPDAD[7:2]+40h. The 40h offset ensures that VPD accesses do not overwrite the PCI 6520 serial EEPROM Configuration data stored in serial EEPROM locations 00h to 3Fh.	Yes	Yes	0
14:8	***Reserved.***	Yes	No	0h
15	**VPD Operation**. Writing 0 generates a Read cycle from the serial EEPROM at the VPD address specified in PVPDAD[7:2]. This bit remains at logic 0 until the serial EEPROM cycle is complete, at which time the bit is set to 1. Data for reads is available in the VPD Data register (PVPDATA; PCI:ECh). Writing 1 generates a Write cycle to the serial EEPROM at the VPD address specified in PVPDAD[7:2]. Remains at logic 1, until the serial EEPROM cycle is completed, at which time the bit is cleared to 0. Place data for writes into the VPD Data register.	Yes	Yes	0

Register 6-82. (PVPDATA; PCI:ECh) VPD Data

Bit	Description	Read	Write	Value after Reset
31:0	**VPD Data (Serial EEPROM Data).** The least significant byte of this register corresponds to the byte of VPD at the address specified by the VPD Address register (PVPDAD[7:2]; PCI:EAh). Data is read from or written to PVPDATA, using standard Configuration accesses.	Yes	Yes	0h

6.1.2.21 PCI-X Capability

Register 6-83. (PCIXCAPID; PCI:F0h) PCI-X Capability ID

Bit	Description	Read	Write	Value after Reset
7:0	**PCI-X Capability ID.** PCI-SIG-issued Capability ID for PCI-X is 07h.	Yes	No	07h

Register 6-84. (PCIX_NEXT; PCI:F1h) PCI-X Next Capability Pointer

Bit	Description	Read	Write	Value after Reset
7:0	**Next_Cap Pointer.** Provides an offset into PCI Configuration space for the location of the next capability in the New Capabilities Linked List. Set to 0h to indicate the end of the Capabilities list.	Yes	No	0h

Register 6-85. (PCIXSSR; PCI:F2h) PCI-X Secondary Status

Bit	Description	Read	Write	Value after Reset
0	**64-Bit Device.** Indicates the PCI 6520 secondary AD interface width. Values: 0 = 32-bit bus data width 1 = 64-bit bus data width	Yes	No	1
1	**133 MHz Capable.** Indicates PCI 6520 secondary interface is capable of 133 MHz operations in PCI-X mode. Values: 0 = Maximum operating frequency is 66 MHz 1 = Maximum operating frequency is 133 MHz ***Note:*** *Hardwired to 1.*	Yes	No	1
2	**Split Completion Discarded.** Set if PCI 6520 discards a Split Completion moving toward the secondary bus, because requester would not accept it. Values: 0 = Split Completion not discarded 1 = Split Completion discarded	Yes	Yes/Clr	0
3	**Unexpected Split Completion.** Set if an Unexpected Split Completion with a Requester ID equal to PCI 6520 secondary Bus Number, Device Number 00h, and Function Number 000b is received on PCI 6520 secondary interface. Values: 0 = No Unexpected Split Completion received 1 = Unexpected Split Completion received	Yes	Yes/Clr	0

Register 6-85. (PCIXSSR; PCI:F2h) PCI-X Secondary Status (Continued)

Bit	Description	Read	Write	Value after Reset
4	**Split Completion Overrun.** Set if PCI 6520 terminates a Split Completion on the secondary bus, with Retry or Disconnect at the next ADB, because PCI 6520 buffers are full. Values: 0 = Bridge accepted all Split Completions 1= Bridge terminated a Split Completion, with a Retry or Disconnect at the next ADB because the bridge buffers are full	Yes	Yes/Clr	0
5	**Split Request Delayed.** Set if PCI 6520 received a request to forward a transaction on the secondary bus, but cannot because there is insufficient space within the limit specified in the PCI-X Downstream Split Transaction register Split Transaction Commitment Limit bits (PCIXDNSTR[31:16]; PCI:FCh). Values: 0 = Bridge did not delay a Split Request 1 = Bridge delayed a Split Request	Yes	Yes/Clr	0
8:6	**Secondary Clock Frequency**. Enables configuration software to determine to which mode (and in PCI-X mode, to which frequency) the PCI 6520 set the secondary bus the last time S_RSTOUT# was asserted. Refer to Table 6-3 for values.	Yes	No	—
15:9	***Reserved.***	Yes	No	0h

Table 6-3. Secondary Clock Frequency Values (PCIXSSR[8:6]; PCI:F2h)

PCIX100MHZ	S_XCAP_IN	PCIXSSR[8:6]	Secondary Bus	Minimum Clock Period
x	Ground	000b	Conventional PCI mode	N/A
x	Pulled Low	001b	66 MHz PCI-X mode	15 ns
1	Pulled Up	010b	100 MHz PCI-X mode	10 ns
0	Pulled Up	011b	133 MHz PCI-X mode	7.5 ns

Note: *Values where PCIXSSR[8]=1 are* ***reserved****.*

Register 6-86. (PCIXBSR; PCI:F4h) PCI-X Bridge Status

Bit	Description	Read	Write	Value after Reset
2:0	**Function Number.** Indicates the number of this function—the number in AD[10:8] of a Type 0 Configuration Transaction address to which this bridge responds. The function uses this number as part of its Requester and Completer IDs (set to 000b). PCI 6520 uses the Bus, Device, and Function Numbers fields to create the Completer ID when responding with a Split Completion to a read of an internal bridge register.	Yes	No	000b
7:3	**Device Number.** Indicates Device Number (the number in AD[15:11] of the address of a Type 0 Configuration transaction) assigned to PCI 6520.	Yes	No	11111b
15:8	**Bus Number.** Additional addresses from which the Primary Bus Number register contents (in the Type 01h Configuration Space header) are read.	Yes	No	—
16	**64-Bit Device.** Indicates the bridge AD bus data width. This bit is the inverse of the DEV64# input. Values: 0 = 32-bit bus data width 1 = 64-bit bus data width	Yes	No	Inverse of DEV64# Input
17	**133 MHz Capability.** Indicates bridge primary interface is capable of 133 MHz operations in PCI-X mode. Values: 0 = Device maximum frequency is 66 MHz 1 = Device maximum frequency is 133 MHz	Yes	No	1
18	**Split Completion Discard.** Set if PCI 6520 discards a Split Completion, because the requester on the primary bus would not accept it. Values: 0 = Split Completion not discarded 1 = Split Completion discarded	Yes	Yes/Clr	0
19	**Unexpected Split Completion.** Set if an Unexpected Split Completion with a Requester ID equal to PCI 6520 Primary Bus, Device, and Function Numbers is received on the bridge primary bus. Values: 0 = No Unexpected Split Completion was received 1 = Unexpected Split Completion was received	Yes	Yes/Clr	0
20	**Split Request Delay.** Set if PCI 6520 terminates a Split Completion on the primary bus, with Retry or Disconnect at the next ADB, because the bridge buffers are full. Used by algorithms that optimize the upstream Split Transaction Commitment Limit register setting. Values: 0 = Bridge accepted all Split Completions 1 = Bridge terminated a Split Completion with Retry or Disconnect	Yes	Yes/Clr	0
21	**Split Completion Overrun.** Set when PCI 6520 receives a request to forward a transaction on the primary bus, but cannot because there is insufficient space within the limit specified in the PCI-X Upstream Split Transaction register Split Transaction Commitment Limit bits (PCIXUPSTR[31:16]; PCI:F8h). Values: 0 = Bridge did not delay a Split Request 1= Bridge delayed a Split Request	Yes	Yes/Clr	0
31:22	***Reserved.***	Yes	No	0h

Register 6-87. (PCIXUPSTR; PCI:F8h) PCI-X Upstream Split Transaction

Bit	Description	Read	Write	Value after Reset
15:0	**Split Transaction Capacity.** PCI 6520 stores Split Completions for Memory Reads in the same buffer as Split Completions for I/O and Configuration Reads and Writes. Indicates the buffer size, in ADQ numbers, for storing Split Completions for Memory Reads for requesters on the secondary bus addressing completers on the primary bus.	Yes	No	32d
31:16	**Split Transaction Commitment Limit or Outstanding ADQ Limit.** Indicates the cumulative Sequence size for PCI-X Memory Read transactions forwarded by PCI 6520 from requesters on the secondary bus addressing completers on the primary bus. Also indicates upstream Split Transaction size of those types the PCI 6520 is allowed to commit to at one time.	Yes	Yes	32d

Note: *PCIXUPSTR controls the bridge buffer behavior for forwarding Split Transactions from a secondary requester to a primary bus completer.*

Register 6-88. (PCIXDNSTR; PCI:FCh) PCI-X Downstream Split Transaction

Bit	Description	Read	Write	Value after Reset
15:0	**Split Transaction Capacity.** PCI 6520 stores Split Completions for Memory Reads in the same buffer as Split Completions for I/O and Configuration Reads and Writes. Indicates the buffer size, in ADQ numbers, for storing Split Completions for Memory Reads for requesters on the primary bus addressing completers on the secondary bus.	Yes	No	32d
31:16	**Split Transaction Commitment Limit or Outstanding ADQ Limit.** Indicates the cumulative Sequence size for PCI-X Memory Read transactions forwarded by PCI 6520 from requesters on the secondary bus addressing completers on the primary bus. Also indicates downstream Split Transaction size of those types the PCI 6520 is allowed to commit to at one time.	Yes	Yes	32d

Note: *PCIXDNSTR controls the bridge buffer behavior for forwarding Split Transactions from a primary bus requester to a secondary bus completer.*

7 SERIAL EEPROM

This section describes information specific to the PCI 6520 serial EEPROM interface and use—access, Autoload mode, and data structure.

7.1 OVERVIEW

Important Note: *Erroneous serial EEPROM data can cause the PCI 6520 to lock the system. Provide an optional switch or jumper to disable the serial EEPROM in board designs.*

The PCI 6520 provides a two-wire interface to a serial EEPROM device. The interface can control an ISSI IS24C02 or compatible part, which is organized as 256 x 8 bits. The serial EEPROM is used to initialize the internal PCI 6520 registers, and alleviates the need for user software to configure the PCI 6520. If a programmed serial EEPROM is connected, the PCI 6520 automatically loads data from the serial EEPROM after P_RSTIN# de-assertion.

The serial EEPROM data structure is defined in Section 7.4.1. The serial EEPROM interface is organized on a 16-bit base in Little Endian format, and the PCI 6520 supplies a 7-bit serial EEPROM Word address.

The following pins are used for the serial EEPROM interface:

- **EEPCLK**—Serial EEPROM clock output
- **EEPDATA**—Serial EEPROM bi-directional serial data

Note: *The PCI 6520 does **not** control the serial EEPROM A0 to A2 address inputs. It can only access serial EEPROM addresses set to 0.*

7.2 SERIAL EEPROM ACCESS

The PCI 6520 can access the serial EEPROM on a Word basis, using the hardware sequencer. Users access one Word data by way of the PCI 6520 Serial EEPROM Control register:

- Serial EEPROM Start/Read/Write Control (EEPCNTRL; PCI:54h)
- Serial EEPROM Address (EEPADDR; PCI:55h)
- Serial EEPROM Data (EEPDATA; PCI:56h)

Before each access, software should check the Auto Mode Cycle in Progress status (EEPCNTRL[0]; PCI:54h, same bit as Start) before issuing the next Start. The following is the general procedure for Read/Write Serial EEPROM accesses:

1. Program the Serial EEPROM Address register (EEPADDR; PCI:55h) with the Word address.
2. **Writes**—Program Word data to the Serial EEPROM Data register (EEPDATA; PCI:56h).

   Reads—Proceed to the next step.
3. **Writes**—Set the Serial EEPROM Command and Start bits (EEPCNTRL[1:0]=11b; PCI:54h, respectively) to start the Serial EEPROM Sequencer.

   Reads—Set the Start bit (EEPCNTRL[1:0]=01b; PCI:54h) to start the Serial EEPROM Sequencer.
4. When the serial EEPROM read/write is complete, as indicated by the bit value of 0 (Serial EEPROM Control register, EEPCNTRL[0]=0; PCI:54h):

   Writes—Data was successfully written to the serial EEPROM.

   Reads—Data was loaded into the Serial EEPROM Data register (EEPDATA; PCI:56h) by the serial EEPROM sequencer.

7.3 SERIAL EEPROM AUTOLOAD MODE AT RESET

Upon PWRGD or P_RSTIN# going high (whichever occurs later), the PCI 6520 autoloads the serial EEPROM data into the internal PCI 6520 registers.

The PCI 6520 initially reads the first offset in the serial EEPROM, which should contain a valid signature value of 1516h. If the signature is correct, register autoload commences immediately commences after reset. During autoload, the PCI 6520 reads sequential words from the serial EEPROM and writes to the appropriate registers. If a blank serial EEPROM is connected, the PCI 6520 stops loading the serial EEPROM contents after reading the first word, as the serial EEPROM's signature is not valid. Likewise, if no serial EEPROM is connected, the PCI 6520 also stops loading the serial EEPROM contents after attempting to read the first word.

7.4 SERIAL EEPROM DATA STRUCTURE

Following reset and the previously described conditions, the PCI 6520 autoloads the registers with serial EEPROM data. Figure 7-1 illustrates the serial EEPROM data structure.

The PCI 6520 accesses the serial EEPROM, one word at a time. **It is important to note that in the Data phase, bit orders are the reverse of that in the Address phase. The PCI 6520 supports only Serial EEPROM Device Address 0.**

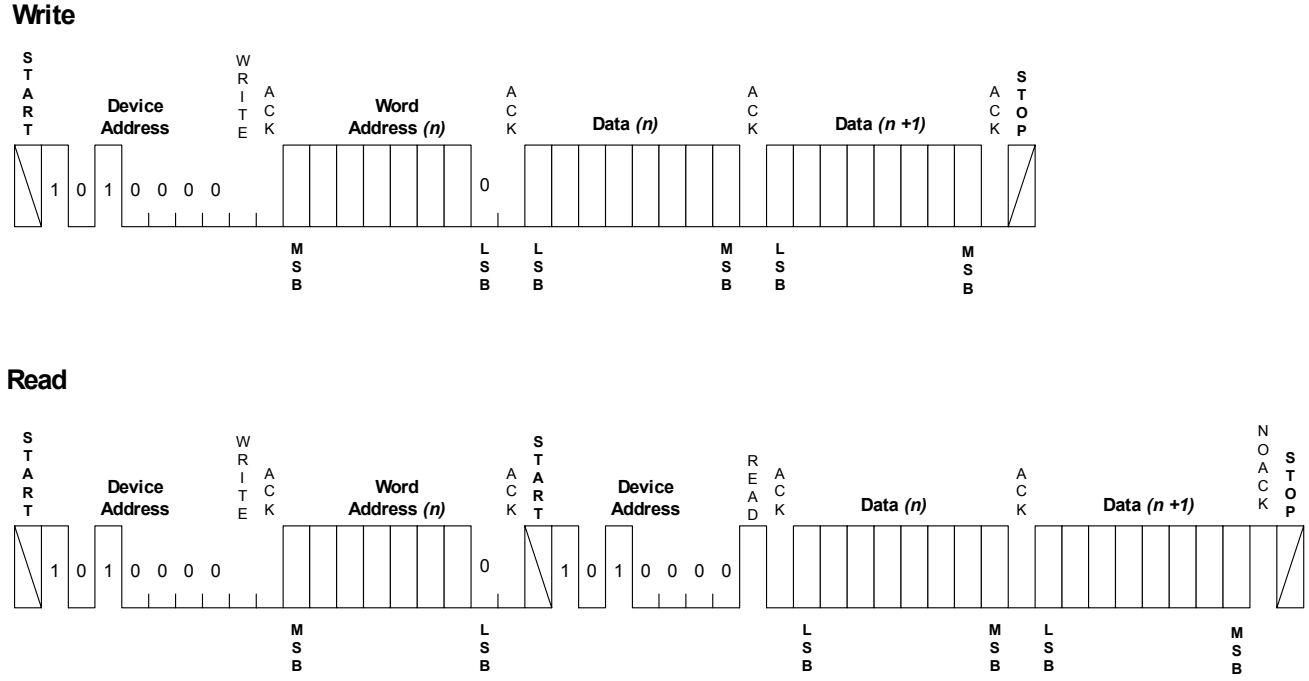


Figure 7-1. Serial EEPROM Data Structure

7.4.1 Serial EEPROM Address and Corresponding PCI 6520 Registers

Table 7-1. Serial EEPROM Address and Corresponding PCI 6520 Registers

Serial EEPROM Byte Address	PCI Configuration Offset	Description
00h – 01h	—	**Serial EEPROM Signature.** Autoload proceeds only if it reads a value of 1516h on the first word loaded. Value: 1516h = Valid signature; otherwise, disables autoloading.
02h	—	**Region Enable.** Enables or disables certain regions of the PCI Configuration space from being loaded from the serial EEPROM. Valid combinations are: Bit 0 = ***Reserved.*** Bits [4:1] = 0000b = Stops autoload at serial EEPROM offset 03h = Group 1. 0001b = Stops autoload at serial EEPROM offset 13h = Group 2. 0011b = Stops autoload at serial EEPROM offset 23h = Group 3. 0111b, 1111b = ***Reserved.*** Other combinations are undefined. Bits [7:5] = ***Reserved.***
03h	—	**Enable Miscellaneous Functions.** Bits [7:0] = ***Reserved.***
End of Group 1		
04h – 05h	00h – 01h	**Vendor ID (PCIIDR[15:0]).**
06h – 07h	02h – 03h	**Device ID (PCIIDR[31:16]).**
08h	—	***Reserved.***
09h	09h	**Class Code.** Contains low byte of Class Code register (PCICCR[7:0]).
0Ah – 0Bh	0Ah – 0Bh	**Class Code Higher Bytes.** Contains upper bytes of Class Code register (PCICCR[23:8]).
0Ch	0Eh	**Header Type (PCIHTR).**
0Dh	09h	***Reserved.***
0Eh – 0Fh	0Ah – 0Bh	***Reserved.***
10h	0Eh	***Reserved.***
11h	0Fh	**Built-In Self Test (BIST) (PCIBISTR).** ***Not supported.*** Set to 0.
12h – 13h	50h	**Internal Arbiter Control (IACNTRL).**
End of Group 2		
14h	44h	**Primary Flow-Through Control (PFTCR).**
15h	45h	**Timeout Control (TOCNTRL).**
16h – 17h	46h – 47h	**Miscellaneous Options (MSCOPT).**
18h	48h	**Primary Initial Prefetch Count (PITLPCNT).**
19h	49h	**Secondary Initial Prefetch Count (SITLPCNT).**
1Ah	4Ah	**Primary Incremental Prefetch Count (PINCPCNT).**
1Bh	4Bh	**Secondary Incremental Prefetch Count (SINCPCNT).**
1Ch	4Ch	**Primary Maximum Prefetch Count (PMAXCNT).**
1Dh	4Dh	**Secondary Maximum Prefetch Count (SMAXCNT).**
1Eh	4Eh	**Secondary Flow-Through Control (SFTCR).**
1Fh	E3h	**Power Management Data (PMCDATA).**
20h – 21h	E0h	**Power Management Control/Status (PMCSR).**
22h – 23h	DEh	**Power Management Capabilities (PMC).**

Table 7-1. Serial EEPROM Address and Corresponding PCI 6520 Registers (Continued)

Serial EEPROM Byte Address	PCI Configuration Offset	Description
End of Group 3		
24h – 25h	2Ch	***Reserved. Must be 0.***
26h – 27h	2Eh	***Reserved. Must be 0.***
End of Group 4		
28h – 3Fh	—	***Reserved. Must be 0.***
End of Group 5		

8 PCI BUS OPERATION

This section describes PCI transactions to which the PCI 6520 responds and those it initiates when operating with one or both of its interfaces in Conventional PCI mode.

8.1 CONVENTIONAL PCI TRANSACTIONS

Table 8-1 lists the Conventional PCI command codes and transaction types to which the PCI 6520 responds and initiates. The *Master* and *Target* columns indicate support for transactions wherein the PCI 6520 initiates transactions as a master, and responds to transactions as a target, on the primary and secondary buses.

Table 8-1. Conventional PCI Transactions

CBE[3:0]#	Transaction Type	Initiates as Master		Responds as Target	
		Primary	Secondary	Primary	Secondary
0000b	Interrupt Acknowledge ***(Not Supported)***	N	N	N	N
0001b	Special Cycle ***(Not Supported)***	Y	Y	N	N
0010b	I/O Read	Y	Y	Y	Y
0011b	I/O Write	Y	Y	Y	Y
0100b	***Reserved***	N	N	N	N
0101b	***Reserved***	N	N	N	N
0110b	Memory Read	Y	Y	Y	Y
0111b	Memory Write	Y	Y	Y	Y
1000b	***Reserved***	N	N	N	N
1001b	***Reserved***	N	N	N	N
1010b	Configuration Read	N	Y	Y	N
1011b	Configuration Write	Type 1	Y	Y	Type 1
1100b	Memory Read Multiple	Y	Y	Y	Y
1101b	Dual Address Cycle (DAC)	Y	Y	Y	Y
1110b	Memory Read Line	Y	Y	Y	Y
1111b	Memory Write and Invalidate	Y	Y	Y	Y

As indicated in Table 8-1, the PCI 6520 does ***not*** support the following Conventional PCI commands—it ignores them and reacts to these commands as follows:

- ***Reserved***—The PCI 6520 does not generate ***reserved*** command codes.
- **Interrupt Acknowledge**—The PCI 6520 never initiates an Interrupt Acknowledge transaction and, as a target, it ignores Interrupt Acknowledge transactions. Interrupt Acknowledge transactions are expected to reside entirely on the primary PCI Bus closest to the host bridge.
- **Special Cycle**—The PCI 6520 does not respond to Special Cycle transactions. To generate Special Cycle transactions on other PCI Buses (downstream or upstream), use a Type 1 Configuration command.
- **Type 0 Configuration Write**—The PCI 6520 does ***not*** generate Type 0 Configuration Write transactions on the primary interface.

8.2 SINGLE ADDRESS PHASE

The PCI 6520 32-bit address uses a single Address phase. This address is driven on AD[31:0], and the bus command is driven on P_CBE[3:0]#.

The PCI 6520 supports only the linear increment Address mode, which is indicated when the lower two Address bits equal 00b. If either of the lower two Address bits is equal to a non-zero value, the PCI 6520 automatically Disconnects the transaction after the first Data transfer.

8.3 DUAL ADDRESS PHASE

The PCI 6520 supports the Dual Address Cycle (DAC) bus command to transfer 64-bit addresses. In DAC transactions, the first Address phase occurs during the initial FRAME# assertion, and the second Address phase occurs one clock later. During the first Address phase, the DAC command is presented on CBE[3:0]#, and the lower 32 bits of the address on AD[31:0]. The second Address phase retains the cycle command on CBE[3:0]#, and the upper 32 bits of the address on AD[31:0]. When a 64-bit master uses DAC, the master must provide the upper 32 bits of the address on AD[63:32] and the command on CBE[7:4]# during the Address phases of both transactions to allow 64-bit targets additional time to decode the transaction.

DACs are used to access locations that are not in the first 4 GB of PCI Memory space. Addresses in the first 4 GB of PCI Memory space always use a Single Address Cycle (SAC).

The PCI 6520 supports DACs in the downstream and upstream directions. The PCI 6520 responds to DACs for the following commands only:

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

The PCI 6520 forwards DACs downstream when their addresses fall within Prefetchable Memory space. DACs originating on the secondary bus, with addresses outside Prefetchable Memory space, are forwarded upstream.

8.4 DEVICE SELECT (DEVSEL#) GENERATION

The PCI 6520 performs positive address decoding when accepting transactions on the primary or secondary bus. The PCI 6520 never subtractively decodes. Medium DEVSEL# timing is used for 33 MHz operation. Slow DEVSEL# timing is used for 66 MHz operation.

8.5 DATA PHASE

Depending on the command type, the PCI 6520 can support multiple Data phase Conventional PCI transactions. Write transactions are treated as Posted Write or Delayed Write transactions.

Table 8-2 lists the forwarding method used for each type of Write operation.

Table 8-2. Write Transaction Forwarding

Transaction Type	Forwarding Method
Memory Write	Posted
Memory Write and Invalidate	
I/O Write	Delayed
Type 1 Configuration Write	

8.5.1 Posted Write Transactions

When the PCI 6520 determines that a Memory Write transaction is to be forwarded across the bridge, the PCI 6520 asserts DEVSEL# with slow timing and TRDY# in the same cycle, provided that sufficient Buffer space is available in the Posted Write Data queue, and that the queue contains fewer than four outstanding Posted transactions. The PCI 6520 can accept one dual-Dword of Write data every PCI Clock cycle (*that is*, no target wait states are inserted). Up to 256 bytes of Posted Write data are stored in internal Posted Write buffers and eventually delivered to the target.

The PCI 6520 continues to accept Write data until one of the following occurs:

- Initiator normally terminates the transaction
- Cache Line boundary or an aligned 4-KB boundary is reached, depending on transaction type
- Posted Write Data buffer fills

When one of the last two events occurs, the PCI 6520 returns a Target Disconnect to the requesting initiator on this Data phase to terminate the transaction.

After the Posted Write transaction is selected for completion, the PCI 6520 requests ownership of the target bus. This can occur while the PCI 6520 is receiving data on the initiator bus. After the PCI 6520 has ownership of the target bus, and the target bus is detected in the idle condition, the PCI 6520 initiates the Write cycle and continues to transfer Write data until all Write data corresponding to that transaction is delivered, or a Target Termination is received. If Write data exists in the queue, the PCI 6520 can drive one dual-Dword of Write data each PCI Clock cycle. If Write data is flowing through the PCI 6520 and the initiator stalls, the PCI 6520 inserts wait states on the target bus if the queue empties.

The PCI 6520 ends the transaction on the target bus when one of the following conditions is met:

- All Posted Write data was delivered to the target
- Target returns a Target Disconnect or Retry (the PCI 6520 starts another transaction to deliver the remaining Write data)
- Target returns a Target Abort (the PCI 6520 discards remaining Write data)

The Master Latency Timer expires, and the PCI 6520 no longer retains the target bus grant (the PCI 6520 starts another transaction to deliver the remaining Write data).

8.5.2 Memory Write and Invalidate Transactions

Memory Write and Invalidate transactions guarantee the transfer of entire cache lines. By default, the PCI 6520 Retries a Memory Write and Invalidate cycle until there is space for one or more cache lines of data in the internal buffers. The PCI 6520 then completes the transaction on the secondary bus as a Memory Write and Invalidate cycle. The PCI 6520 can also be programmed to accept Memory Write and Invalidate cycles under the same conditions as normal Memory Writes. In this case, if the Write buffer fills before an entire cache line is transferred, the PCI 6520 Disconnects and completes the Write cycle on the secondary bus as a normal Memory Write cycle by way of the Miscellaneous Options register Memory Write and Invalidate Control bit (MSCOPT[12]; PCI:46h). The PCI 6520 Disconnects Memory Write and Invalidate commands at Aligned Cache Line boundaries. The Cache Line Size register (PCICLSR; PCI:0Ch) cache line size value provides the number of Dwords in a cache line. For the PCI 6520 to generate Memory Write and Invalidate transactions, this cache line size value must be written to a value of 08h, 10h, or 20h Dwords. If an invalid cache line size is programmed, wherein the value is 0, not a power of two, or greater than 20h Dwords, the PCI 6520 sets the cache line size to the minimum value of 08h. The PCI 6520 always Disconnects on the Cache Line boundary.

When the Memory Write and Invalidate transaction is Disconnected before a Cache Line boundary is reached (typically because the Posted Write Data buffer fills), the transaction is converted to a Memory Write transaction.

8.5.3 Delayed Write Transactions

A Delayed Write transaction forwards I/O Write and Type 1 Configuration cycles by way of the PCI 6520, and is limited to a single Dword Data transfer.

When a Write transaction is first detected on the initiator bus, the PCI 6520 claims the access and

returns a Target Retry to the initiator. During the cycle, the PCI 6520 samples the Bus Command, Address, and Address Parity bits. The PCI 6520 also samples the first data Dword, Byte Enable bits, and data parity. Cycle information is placed into the Delayed Transaction queue if there are no other existing Delayed transactions with the same cycle information, and if the Delayed Transaction queue is not full. When the PCI 6520 schedules a Delayed Write transaction to be the next cycle to complete based on its ordering constraints, the PCI 6520 initiates the transaction on the target bus. The PCI 6520 transfers the Write data to the target.

If the PCI 6520 receives a Target Retry in response to the Write transaction on the target bus, the PCI 6520 continues to repeat the Write transaction until the Data transfer is complete, or an error condition is encountered. If the PCI 6520 is unable to deliver Write data after 2^{24} attempts (programmable through the Timeout Control register Maximum Retry Counter Control bits, TOCNTRL[2:0]; PCI:45h), the PCI 6520 ceases further write attempts and returns a Target Abort to the initiator. The Delayed transaction is removed from the Delayed Transaction queue.

The PCI 6520 also asserts P_SERR# if the Command register P_SERR# Enable bit is set (PCICR[8]=1; PCI:04h). When the initiator repeats the same Write transaction (same command, address, Byte Enable bits, and data), after the PCI 6520 has completed data delivery and retains all complete cycle information in the queue, the PCI 6520 claims the access and returns TRDY# to the initiator, indicating that the Write data was transferred. If the initiator requests multiple Dwords, the PCI 6520 asserts STOP#, in conjunction with TRDY#, to signal a Target Disconnect. Only those bytes of Write data with valid Byte Enable bits are compared. If any Byte Enable bits are disabled (driven high), the corresponding byte of Write data is ***not*** compared.

If the initiator repeats the Write transaction before the data is transferred to the target, the PCI 6520 returns a Target Retry to the initiator. The PCI 6520 continues to return a Target Retry to the initiator until Write data is delivered to the target or an error condition is encountered. When the Write transaction is repeated, the PCI 6520 does ***not*** make a new entry into the Delayed Transaction queue.

The PCI 6520 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 four values, selectable through the primary and secondary Bridge Control register Master Timeout bits (BCNTRL[8:9]; PCI:3Eh), as well as the Timeout Control register Master Timeout Divider bits (TOCNTRL[7:4]; PCI:45h). If the Discard Timer expires before the Write cycle is Retried, the PCI 6520 discards the Delayed Write transaction from the Delayed Transaction queue. The PCI 6520 also conditionally asserts P_SERR#.

8.5.4 Write Transaction Address Boundaries

The PCI 6520 imposes internal Address boundaries when accepting Write data. The Aligned Address boundaries are used to prevent the PCI 6520 from continuing a transaction over a device Address boundary and to provide an upper limit on maximum latency. When the Aligned Address boundaries are reached (per conditions listed in Table 8-3), the PCI 6520 returns a Target Disconnect to the initiator.

8.5.5 Buffering Multiple Write Transactions

The PCI 6520 continues to accept Posted Memory Write transactions if space for at least 1 Dword of data in the Posted Write Data buffer remains and there are fewer than four outstanding Posted Memory Write cycles. If the Posted Write Data buffer fills before the initiator terminates the Write transaction, the PCI 6520 returns a Target Disconnect to the initiator.

Delayed Write transactions are posted when one or more open entries exist in the Delayed Transaction queue. The PCI 6520 can queue up to four Posted Write transactions and four Delayed transactions in both the downstream and upstream directions.

8.5.6 Read Transactions

Delayed Read forwarding is used for all Read transactions that cross the PCI 6520.

Delayed Read transactions are treated as prefetchable or non-prefetchable.

Table 8-4 delineates the read behavior (prefetchable or non-prefetchable) for each type of Read operation.

Table 8-3. Write Transaction Disconnect Address Boundaries

Transaction Type	Condition	Aligned Address Boundary
Delayed Write	All	Disconnects after one Data transfer
Posted Memory Write	Memory Write Disconnect Control Bit = 0[1]	4-KB Aligned Address boundary
	Memory Write Disconnect Control Bit = 1[1]	Disconnects at Cache Line boundary
Posted Memory Write and Invalidate	Cache Line Size = 8h	8h-Dword aligned Address boundary
	Cache Line Size = 10h	10h-Dword aligned Address boundary
	Cache Line Size = 12h	12h-Dword aligned Address boundary

1. Memory Write Disconnect Control bit is located in the Chip Control register in Configuration space (CCNTRL[1]; PCI:40h).

Table 8-4. Read Transaction Prefetching

Transaction Type	Read Behavior
I/O Read	Never prefetches
Configuration Read	
Memory Read	Downstream—Prefetches if address is in prefetchable space Upstream—Prefetches if prefetch disable is off (default)
Memory Read Line	Always prefetches if request is for more than one Data transfer
Memory Read Multiple	

8.5.7 Prefetchable Read Transactions

A Prefetchable Read transaction is a Read transaction wherein the PCI 6520 performs speculative DWORD reads, transferring data from the target before the initiator requests the data. This behavior allows a Prefetchable Read transaction to consist of multiple Data transfers. Only the first Byte Enable bits can be forwarded. The PCI 6520 enables all Byte Enable bits of subsequent transfers.

Prefetchable behavior is used for Memory Read Line and Memory Read Multiple transactions, as well as Memory Read transactions that fall into Prefetchable Memory space.

The amount of prefetched data depends on the transaction type. The amount of prefetching may also be affected by the amount of free space in the PCI 6520 Read FIFO and by the Read Address boundaries encountered. In addition, there are several PCI 6520-specific registers that can be used to optimize read prefetch behavior.

Prefetching should not be used for those Read transactions that cause side effects on the target device (*that is*, Control and Status registers, FIFOs, and so forth). The target device BARs indicate whether a Memory Address region is prefetchable.

8.5.8 Non-Prefetchable Read Transactions

A Non-Prefetchable Read transaction is a Read transaction issued by the initiator into a non-prefetchable region. The transaction is used for I/O and Configuration Read transactions, as well as for Memory Reads from Non-Prefetchable Memory space. In this case, the PCI 6520 requests only 1 Dword from the target and Disconnects the initiator after delivery of the first Dword of Read data.

Use Non-Prefetchable Read transactions for regions in which extra Read transactions could have side effects (*such as* in FIFO memory or the Control registers). If it is important to retain the Byte Enable bit values during the Data phase of cycles forwarded across the bridge, use Non-Prefetchable Read transactions. If these locations are mapped into Memory space, use the Memory Read command and map the target into Non-Prefetchable (Memory-Mapped I/O) Memory space to utilize non-prefetching behavior.

8.5.9 Read Prefetch Address Boundaries

The PCI 6520 imposes internal Read Address boundaries on read prefetching. The PCI 6520 uses the Address boundary to calculate the initial amount of prefetched data. During Read transactions to Prefetchable regions, the PCI 6520 prefetches data until it reaches one of these aligned Address boundaries, unless the target signals a Target Disconnect before reaching the Read Prefetch boundary. After reaching the Aligned Address boundary, the PCI 6520 may optionally continue prefetching data, depending on certain conditions. (Refer to Section 18, "PCI Flow-Through Optimization.") When finished transferring Read data to the initiator, the PCI 6520 returns a Target Disconnect with the last Data transfer, unless the initiator completes the transaction before delivering all the prefetched Read data. Remaining prefetched data is discarded.

Prefetchable Read transactions in Flow-Through mode prefetch to the nearest aligned 4-KB Address boundary, or until the initiator de-asserts FRAME#.

Table 8-5 delineates the Read Prefetch Address boundaries for Read transactions during Non-Flow-Through mode.

Table 8-5. Read Prefetch Address Boundaries

Transaction Type	Address Space	Prefetch Aligned Address Boundary
Configuration Read	—	1 Dword (No Prefetch)
I/O Read		
Memory Read	Non-Prefetchable	
Memory Read	Prefetchable	Configured by way of Prefetch Count registers
Memory Read Line		
Memory Read Multiple		

8.5.10 Delayed Read Requests

The PCI 6520 treats all Read transactions as Delayed Read transactions (*that is*, the Read request from the initiator is posted into a Delayed Transaction queue). Read data from the target is placed into the Read Data queue directed toward the initiator bus interface and transferred to the initiator when the initiator repeats the Read transaction.

When the PCI 6520 accepts a Delayed Read request, it first samples the Read address, Read bus command, and address parity. When IRDY# is asserted, the PCI 6520 samples the Byte Enable bits for the first Data phase. This information is entered into the Delayed Transaction queue. The PCI 6520 terminates the transaction by signaling a Target Retry to the initiator. Upon receiving the Target Retry, the initiator must to continue to repeat the same Read transaction until at least one Data transfer completes, or until it receives a target response other than a Target Retry (Master or Target Abort).

8.5.11 Delayed Read Completion with Target

When a Delayed Read request is scheduled to be executed, the PCI 6520 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 non-prefetchable, the PCI 6520 drives the captured Byte Enable bits during the next cycle. If the transaction is a Prefetchable Read transaction, the PCI 6520 drives the captured (first) Byte Enable bits, followed by 0 for the subsequent Data phases. If the PCI 6520 receives a Target Retry in response to the Read transaction on the target bus, it repeats the Read transaction until at least one Data transfer completes or it encounters an error condition. If the transaction is terminated by way of a normal Master Termination or Target Disconnect after at least one Data transfer is complete, the PCI 6520 does ***not*** initiate further attempts to read additional data.

If the PCI 6520 is unable to obtain Read data from the target after 2^{24} attempts (default), the PCI 6520 ceases further read attempts and returns a Target Abort to the initiator. The Delayed transaction is removed from the Delayed Transaction queue. The PCI 6520 also asserts P_SERR# if the Command register P_SERR# Enable bit is set (PCICR[8]=1; PCI:04h).

After receiving DEVSEL# and TRDY# from the target, the PCI 6520 transfers the data stored in the internal Read FIFO, then terminates the transaction. The PCI 6520 can accept 1 Dword/Qword of Read data during each PCI Clock cycle—no master wait states are inserted. The number of Dwords/Qwords transferred during a Delayed Read transaction depends on the conditions delineated in Table 8-5 (assuming no Target Disconnect is received).

8.5.12 Delayed Read Completion on Initiator Bus

When the Delayed Read transaction completes on the target bus, the Delayed Read data is at the head of the Read Data queue. When all ordering constraints with Posted Write transactions are satisfied, the PCI 6520 transfers the data to the initiator when the initiator repeats the transaction. For Memory Read transactions, the PCI 6520 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. The PCI 6520 returns a Target Disconnect along with the transfer of the last Dword of Read data to the initiator. If the PCI 6520 initiator terminates the transaction before all Read data is transferred, the remaining Read data in the Data buffers is discarded.

When the master repeats the transaction and starts transferring prefetchable Read data from the Data buffers while the Read transaction on the target bus is 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 from the target to the initiator through the Data buffers, long Read bursts can be sustained. In this case, the Read transaction is allowed to continue until the initiator terminates the transaction, an aligned 4-KB Address boundary is reached, or the buffer fills, whichever occurs first. When the buffer empties, the PCI 6520 reflects the stalled condition to the initiator by de-asserting TRDY# for a maximum of eight clock periods until more Read data is available; otherwise, the PCI 6520 Disconnects the cycle. When the initiator terminates the transaction, the PCI 6520 de-assertion

of FRAME# on the initiator bus is forwarded to the target bus. Any remaining Read data is discarded.

The PCI 6520 implements a Discard Timer (BCNTRL[9 or 8]; PCI:3Eh) 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. If the initiator does not repeat the Read transaction before the Discard Timer expires, the PCI 6520 discards the Read transaction, discards the Read data from its queues, and conditionally asserts P_SERR#.

The PCI 6520 has the capability to post multiple Delayed Read requests, up to a maximum of four in both directions. If an initiator starts a Read transaction that matches the Address and Read command of a queued Read transaction, the current Read command is not stored because it is contained in the Delayed Transaction queue.

8.5.13 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 registers are accessible only in Configuration space.

In addition to accepting Configuration transactions for initialization of its own Configuration space, the PCI 6520 forwards Configuration transactions for device initialization in hierarchical PCI Bus systems, as well as Special Cycle generation.

To support hierarchical PCI Bus systems, Type 0 and Type 1 Configuration transactions are specified.

Type 0 Configuration transactions are issued when the intended target resides on the same PCI Bus as the initiator. Type 0 Configuration transactions are identified by the Configuration command and the lowest two bits of the address are set to 00b.

Type 1 Configuration transactions are issued when the intended target resides on another PCI Bus, or a Special Cycle is to be generated on another PCI Bus. Type 1 Configuration commands are identified by the Configuration command and the lowest two Address bits are set to 01b.

The Register Number is found in both Type 0 and Type 1 formats and provides 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 multi-function device is to be accessed. For single-function devices, this value is not decoded. Type 1 Configuration transaction addresses also include five bits, designating the Device Number that identifies the target PCI Bus device to be accessed. In addition, the Bus Number in Type 1 transactions specifies the target PCI Bus.

8.5.14 PCI 6520 Type 0 Access

Configuration space is accessed by a Type 0 Configuration transaction on the primary interface. Configuration space is ***not*** accessible from the secondary bus. The PCI 6520 responds to a Type 0 Configuration transaction by asserting P_DEVSEL# when the following conditions are met during the Address phase:

- Bus command is a Configuration Read or Write transaction.
- Lower two Address bits on P_AD[1:0] must be 01b.
- P_IDSEL must be asserted.
- PCI 6520 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 Configuration space do not have side effects, all bytes in the requested Dword are returned, regardless of the Byte Enable bit values.
- Type 0 Configuration Read and Write transactions do not use data buffers (*that is*, these transactions are immediately completed, regardless of the Data buffers state).

The PCI 6520 ignores all Type 0 transactions initiated on the secondary interface.

8.5.15 Type 1-to-Type 0 Translation

Type 1 Configuration transactions are specifically used for device configuration in a hierarchical PCI Bus system. A PCI-to-PCI bridge 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.

The PCI 6520 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. The PCI 6520 must convert the Configuration command to a Type 0 format, enabling the secondary bus device to respond to the command. Type 1-to-Type 0 translations are performed only in the downstream direction (*that is*, the PCI 6520 generates a Type 0 transaction only on the secondary bus, and never on the primary bus).

The PCI 6520 responds to a Type 1 Configuration transaction and translates the transaction into a Type 0 transaction on the secondary bus when the following conditions are met during the Address phase:

- Lower two Address bits on P_AD[1:0] are 01b
- Bus Number in address field P_AD[23:16] is equal to the Secondary Bus Number register value in Configuration space (PCISBNO; PCI:19h)
- Bus command on P_CBE[3:0]# is a Configuration Read or Write transaction

When translating a Type 1 transaction to a Type 0 transaction on the secondary interface, the PCI 6520 performs the following translations to the address:

- Sets the lower two Address bits on S_AD[1:0] to 00b
- Decodes the Device Number and drives the bit pattern specified in Table 8-6 on S_AD[31:16] for the purpose of asserting the device's IDSEL signal
- Sets S_AD[15:11] to 0h
- Leaves the Function and Register Number fields unchanged

The PCI 6520 asserts unique address lines, based on the Device Number. These address lines may be used as secondary IDSEL signals. Address line mapping depends on the Device Number in the Type 1 Address bits, P_AD[15:11]. The PCI 6520 uses the mapping presented in Table 8-6.

The PCI 6520 can assert up to 16 unique address lines to be used as secondary IDSEL signals for up to 16 secondary bus devices, for Device Numbers ranging from 0 to 15. Because of the PCI Bus electrical loading constraints, more than 16 IDSEL signals should not be necessary. However, if more than 15 device numbers are needed, an external method of generating IDSEL lines must be used, and the upper Address bits are ***not*** asserted. The Configuration transaction is translated and passed from primary-to-secondary bus. If an IDSEL pin is not asserted to a secondary device, the transaction terminates in a Master Abort.

The PCI 6520 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. When Type 1-to-Type 0 Configuration cycles are forwarded, address stepping is used, and a valid address is driven on the bus before FRAME# assertion. Type 0 Configuration address stepping is programmable through the Miscellaneous Options register Address Step Control bits (MSCOPT[6:4]; PCI:46h).

Table 8-6. Device Number to IDSEL S_AD Pin Mapping

Device Number	P_AD[15:11]	Secondary IDSEL S_AD[31:16]	S_AD Bit
0h	00000b	0000_0000_0000_0001b	16
1h	00001b	0000_0000_0000_0010b	17
2h	00010b	0000_0000_0000_0100b	18
3h	00011b	0000_0000_0000_1000b	19
4h	00100b	0000_0000_0001_0000b	20
5h	00101b	0000_0000_0010_0000b	21
6h	00110b	0000_0000_0100_0000b	22
7h	00111b	0000_0000_1000_0000b	23
8h	01000b	0000_0001_0000_0000b	24
9h	01001b	0000_0010_0000_0000b	25
10h	01010b	0000_0100_0000_0000b	26
11h	01011b	0000_1000_0000_0000b	27
12h	01100b	0001_0000_0000_0000b	28
13h	01101b	0010_0000_0000_0000b	29
14h	01110b	0100_0000_0000_0000b	30
15h	01111b	1000_0000_0000_0000b	31
Special Cycle	1XXXXb	0000_0000_0000_0000b	—

8.5.16 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.

When the PCI 6520 detects a Type 1 Configuration transaction intended for a PCI Bus downstream from the secondary bus, the PCI 6520 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. Downstream Type 1-to-Type 1 forwarding occurs when the following conditions are met during the Address phase:

- Lower two Address bits on AD[1:0] are equal to 01b
- Bus Number falls in the range defined by the lower limit (exclusive) in the Secondary Bus Number register (PCISBNO; PCI:19h) and upper limit (inclusive) in the Subordinate Bus Number register (PCISUBNO; PCI:1Ah)
- Bus command is a Configuration Read or Write transaction

The PCI 6520 also supports Type 1-to-Type 1 upstream Configuration Write transaction forwarding to support upstream Special Cycle generation. A Type 1 Configuration command is forwarded upstream when the following conditions are met:

- Lower two Address bits on AD[1:0] are equal to 01b
- Bus Number falls outside the range defined by the lower limit (inclusive) in the Secondary Bus Number register (PCISBNO; PCI:19h) and upper limit (inclusive) in the Subordinate Bus Number register (PCISUBNO; PCI:1Ah)
- Device Number in Address bits AD[15:11] is equal to 11111b
- Function Number in Address bits AD[10:8] is equal to 111b
- Bus command is a Configuration Write transaction
- PCI 6520 forwards Type 1-to-Type 1 Configuration Write transactions as Delayed transactions, limited to a single Data transfer

8.5.17 Special Cycles

The Type 1 configuration mechanism is used to generate Special Cycle transactions in hierarchical PCI systems. Special Cycle transactions are ignored by operating 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 downstream or upstream direction.

The PCI 6520 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:

- Lower two Address bits on AD[1:0] are equal to 01b
- Device Number in Address bits AD[15:11] is equal to 11111b
- Function Number in Address bits AD[10:8] is equal to 111b
- Register number in Address bits AD[7:2] is equal to 0h
- Bus Number is equal to the Secondary Bus Number register value in Configuration space (PCISBNO; PCI:19h) for downstream forwarding, or equal to the Primary Bus Number register value in Configuration space (PCIPBNO; PCI:18h) for upstream forwarding
- Bus command on the initiator CBE bus is a Configuration Write command

When the PCI 6520 initiates a transaction on the target interface, the bus command is changed from Configuration Write to Special Cycle. The address and data are forwarded, unchanged. Devices that use Special Cycle 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 because Special Cycles complete as Master Aborts. After the transaction is completed on the target bus, through Master Abort condition detection, the PCI 6520 responds with TRDY# to the next attempt of the Configuration transaction from the initiator. If more than one Data transfer is requested, the PCI 6520 responds with a Target Disconnect operation during the first Data phase.

8.6 CONVENTIONAL PCI MODE TRANSACTION TERMINATION

This subsection describes how the PCI 6520 returns transaction termination conditions to the initiator.

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

- **Normal Termination**—Occurs when the initiator de-asserts FRAME# at the beginning of the last Data phase, and de-asserts IRDY# at the end of the last Data phase in conjunction with TRDY# or STOP# assertion from the target.
- **Master Abort**—Occurs when no target response is detected. When the initiator does not detect the DEVSEL# signal from the target within five Clock cycles after asserting FRAME#, the initiator terminates the transaction with a Master Abort. If FRAME# is asserted, the initiator de-asserts FRAME# on the next cycle, then de-asserts IRDY# on the following cycle. IRDY# must be asserted in the same cycle in which FRAME# is de-asserted. If FRAME# was de-asserted, IRDY# can be de-asserted on the next Clock cycle following Master Abort condition detection.

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

- **Normal Termination**—TRDY# and DEVSEL# are asserted in conjunction with FRAME# de-assertion and IRDY# assertion.
- **Target Retry**—STOP# and DEVSEL# are asserted without TRDY# during the first Data phase. No data transfers during the transaction. This transaction must be repeated.
- **Target Disconnect (with Data transfer)**—DEVSEL# and STOP# are asserted with TRDY#. Indicates that this is the last Data transfer of the transaction.
- **Target Disconnect (without Data transfer)**—STOP# and DEVSEL# are asserted without TRDY# after previous Data transfers. Indicates that no further Data transfers are made during this transaction.
- **Target Abort**—STOP# is asserted without DEVSEL# and TRDY#. Indicates that the target is never able to complete this transaction. DEVSEL# must be asserted for at least one cycle during the transaction before the Target Abort is signaled.

8.6.1 PCI 6520-Initiated Master Termination

As an initiator, the PCI 6520 uses normal termination if DEVSEL# is returned by the target within five Clock cycles of PCI 6520 FRAME# assertion on the target bus. In this case, the PCI 6520 terminates a transaction when the following conditions are met:

- During Delayed Write transactions, a single Dword/Qword is delivered.
- During Non-Prefetchable Read transactions, a single Dword/Qword is transferred from the target.
- During Prefetchable Read transactions, a Prefetch boundary is reached.
- For Posted Write transactions, all Write data for the transaction is transferred from Data buffers to the target.
- For Burst transfers (*except* Memory Write and Invalidate transactions), the Master Latency Timer expires and the PCI 6520 bus grant is de-asserted.
- Target terminates the transaction with a Retry, Disconnect, or Target Abort.
- If the PCI 6520 is delivering Posted Write data when it terminates the transaction because the Master Latency Timer expired, the PCI 6520 initiates another transaction to deliver the remaining Write data. The transaction address is updated to reflect the address of the current Dword to be delivered.

If the PCI 6520 is delivering Posted Write data when it terminates the transaction because the Master Latency Timer expires, the PCI 6520 initiates another transaction to deliver the remaining Write data. The Transaction address is updated to reflect the current DWORD address to be delivered.

If the PCI 6520 is prefetching Read data when it terminates the transaction because the Master Latency Timer expired, the PCI 6520 does ***not*** repeat the transaction to obtain additional data.

8.6.2 Master Abort Received by PCI 6520

If the initiator initiates a transaction on the target bus and does not detect DEVSEL# returned by the target within five Clock cycles of FRAME# assertion, the PCI 6520 terminates the transaction, as specified in the Bridge Control register Master Abort Mode bit (BCNTRL[5]; PCI:3Eh).

For Delayed Read and Write transactions, the PCI 6520 can assert TRDY# and return FFFF_FFFFh for reads, or return a Target Abort. SERR# is also optionally asserted.

When a Master Abort is received in response to a Posted Write transaction, the PCI 6520 discards the Posted Write data and makes no further attempts to deliver the data. The PCI 6520 sets the Status register Received Master Abort bit when the Master Abort is received on the primary bus (PCISR[13]=1; PCI:06h), or the Secondary Status register Received Master Abort bit when the Master Abort is received on the secondary interface (PCISSR [13]=1; PCI:1Eh).

When the Master Abort Mode bit is set and a Master Abort is detected in response to a Posted Write transaction, the PCI 6520 also asserts P_SERR#, if enabled (PCICR[8]=1; PCI:04h), but not disabled by the device-specific P_SERR# disable for Master Aborts that occur during Posted Write transactions. (Refer to Table 8-7.)

8.6.3 Target Termination Received by PCI 6520

When the PCI 6520 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 FRAME# de-assertion)
- Target Retry
- Target Disconnect
- Target Abort

The PCI 6520 controls these terminations using various methods, depending on the type of transaction performed.

Table 8-7. P_SERR# Assertion Requirements in Response to Master Abort on Posted Write

Description	Bit
Received Master Abort	PCISR[13]=1; PCI:06h
P_SERR# Enable	PCICR[8]=1; PCI:04h
Master Abort on Posted Write	PSERRED[4]=0; PCI:64h

8.6.3.1 Posted Write Target Termination Response

When the PCI 6520 initiates a Posted Write transaction, the Target Termination ***cannot*** be returned to the initiator. Table 8-8 delineates the response to each type of Target Termination that occurs during a Posted Write transaction.

When a Target Retry or Disconnect is returned and Posted Write data associated with that transaction remains in the Write buffers, the PCI 6520 initiates another Write transaction to attempt to deliver the remaining 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 current Dword address. 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, the PCI 6520 uses the Memory Write command to deliver the remaining Write data because less than a cache line is transferred in the subsequent Write transaction attempt.

After the PCI 6520 makes 2^{24} write attempts and fails to deliver all Posted Write data associated with that transaction, the PCI 6520 asserts P_SERR#, if enabled in the Command register, ***and*** the device-specific P_SERR# Disable bit for this condition is ***not*** set. (Refer to Table 8-9.) The Write data is discarded.

Table 8-8. Response to Posted Write Target Termination

Target Termination	Response
Normal	No additional action.
Target Retry	Repeats Write transaction to target.
Target Abort	Sets target interface Status register Received Target Abort bit (primary—PCISR[12]=1; PCI:06h, secondary—PCISSR[12]=1; PCI:1Eh). Asserts P_SERR#, if enabled, and sets the Primary Status register Signaled System Error bit (PCICR[8]=1; PCI:04h and PCISR[14]=1; PCI:06h, respectively).

Table 8-9. P_SERR# Assertion Requirements in Response to Posted Write Parity Error

Description	Bit
P_SERR# Enable	PCICR[8]=0; PCI:04h
Posted Write Parity Error	PSERRED[1]=0; PCI:64h

8.6.3.2 Delayed Write Target Termination Response

When the PCI 6520 initiates a Delayed Write transaction, the type of Target Termination received from the target can be returned to the initiator. Table 8-10 delineates the response to each type of Target Termination that occurs during a Delayed Write transaction. The PCI 6520 repeats a Delayed Write transaction until the PCI 6520:

- Completes at least one Data transfer
- Receives a Master Abort
- Receives a Target Abort

The PCI 6520 makes 2^{24} write attempts (default), resulting in a response of Target Retry. After the PCI 6520 makes 2^{24} attempts of the same Delayed Write transaction on the target bus, the PCI 6520 asserts P_SERR# if the Command register P_SERR# Enable bit is set and the implementation-specific P_SERR# Disable bit for this condition is not set. (Refer to Table 8-11.) The PCI 6520 stops initiating transactions in response to that Delayed Write transaction and the Delayed Write request is discarded. Upon a subsequent Write transaction attempt by the initiator, the PCI 6520 returns a Target Abort.

Table 8-10. Response to Delayed Write Target Termination

<table>
<tr><th>Target Termination</th><th colspan="4">Response</th></tr>
<tr><td>Normal</td><td colspan="4">Returns Disconnect to initiator with first Data transfer only if multiple Data phases are requested.</td></tr>
<tr><td>Target Retry</td><td colspan="4">Returns Target Retry to initiator. Continue write attempts to target.</td></tr>
<tr><td>Target Disconnect</td><td colspan="4">Returns Disconnect to initiator with first Data transfer only if multiple Data phases are requested.</td></tr>
<tr><td rowspan="3">Target Abort</td><td colspan="4">Returns Target Abort to initiator.
Sets target interface Status register Received Target Abort bit.
Sets initiator interface Status register Signaled Target Abort bit.</td></tr>
<tr><td>Initiator (Primary Bus)</td><td>Target (Secondary Bus)</td><td>Initiator (Secondary Bus)</td><td>Target (Primary Bus)</td></tr>
<tr><td>PCISR[11]=1;
PCI:06h</td><td>PCISSR[12]=1;
PCI:1Eh</td><td>PCISR[12]=1;
PCI:06h</td><td>PCISSR[11]=1;
PCI:1Eh</td></tr>
</table>

Table 8-11. P_SERR# Assertion Requirements in Response to Delayed Write

Description	Bit
P_SERR# Enable	PCICR[8]=1; PCI:04h
Delayed Configuration or I/O Write Non-Delivery	PSERRED[5]=0; PCI:64h

8.6.3.3 Delayed Read Target Termination Response

When the PCI 6520 initiates a Delayed Read transaction, the abnormal target responses can be returned to the initiator. Other target responses depend on the amount of data the initiator requests. Table 8-12 delineates the response to each type of Target Termination that occurs during a Delayed Read transaction.

The PCI 6520 repeats a Delayed Read transaction until the PCI 6520:

- Completes at least one Data transfer
- Receives a Master Abort
- Receives a Target Abort
- Produces 2^{24} read attempts, resulting in a response of Target Retry

After the PCI 6520 produces 2^{24} attempts of the same Delayed Read transaction on the target bus, the PCI 6520 asserts P_SERR# if the Command register P_SERR# Enable bit is set and the implementation-specific P_SERR# Disable bit for this condition is ***not*** set. (Refer to Table 8-13.) The PCI 6520 stops initiating transactions in response to that Delayed Read transaction, and the Delayed Read request is discarded. Upon a subsequent Read transaction attempt by the initiator, the PCI 6520 returns a Target Abort.

Table 8-12. Response to Delayed Read Target Termination

<table>
<tr><th>Target Termination</th><th colspan="4">Response</th></tr>
<tr><td>Normal</td><td colspan="4">If prefetchable, Target Disconnects only if initiator requests more data than read from target. If non-prefetchable, Target Disconnects on first Data phase.</td></tr>
<tr><td>Target Retry</td><td colspan="4">Re-initiates Read transaction to target.</td></tr>
<tr><td>Target Disconnect</td><td colspan="4">If initiator requests more data than read from target, returns Target Disconnect to initiator.</td></tr>
<tr><td rowspan="3">Target Abort</td><td colspan="4">Returns Target Abort to initiator.
Sets target interface Status register Received Target Abort bit.
Sets initiator interface Status register Signaled Target Abort bit.</td></tr>
<tr><td>Initiator (Primary Bus)</td><td>Target (Secondary Bus)</td><td>Initiator (Secondary Bus)</td><td>Target (Primary Bus)</td></tr>
<tr><td>PCISR[11]=1;
PCI:06h</td><td>PCISSR[12]=1;
PCI:1Eh</td><td>PCISR[12]=1;
PCI:06h</td><td>PCISSR[11]=1;
PCI:1Eh</td></tr>
</table>

Table 8-13. P_SERR# Assertion Requirements in Response to Delayed Read

Description	Bit
P_SERR# Enable	PCICR[8]=1; PCI:04
Delayed Read-No Data from Target	PSERRED[6]=0; PCI:64h

8.6.4 PCI 6520-Initiated Target Termination

The PCI 6520 can return a Target Retry, Disconnect, or Abort to an initiator for reasons other than detection of that condition at the target interface.

8.6.4.1 Target Retry

When it cannot accept Write data or return Read data as a result of internal conditions, the PCI 6520 returns a Target Retry to the initiator when any of the following conditions are met:

- **Delayed Write Transactions**
  - Transaction is in the process of entering the Delayed Transaction queue.
  - Transaction has entered the Delayed Transaction queue, but target response has not been received.
  - Target response was received, but the Posted Memory Write Ordering rule prevents the cycle from completing.
  - Delayed Transaction queue is full; therefore, transaction ***cannot*** be queued.
  - Transaction with the same address and command was queued.
  - Locked sequence is being propagated across the PCI 6520, and the Write transaction is not a Locked transaction.
  - Target bus is locked and the Write transaction is a Locked transaction.
- **Delayed Read Transactions**
  - Transaction is in the process of entering the Delayed Transaction queue.
  - Read request was queued, but Read data is not yet available.
  - Data was read from the target, but the data is not at the head of the Read Data queue, or a Posted Write transaction precedes it.
  - Delayed Transaction queue is full, and the transaction ***cannot*** be queued.
  - Delayed Read request with the same address and bus command was queued.
  - Locked sequence is being propagated across the PCI 6520, and the Read transaction is not a Locked transaction.
  - Target bus is locked and the Read transaction is a Locked transaction.
- **Posted Write Transactions**
  - Posted Write Data buffer does not contain sufficient space for the address and at least two Qwords of Write data.
  - Locked sequence is being propagated across the PCI 6520, and the Write transaction is not a Locked transaction.

When a Target Retry is returned to a Delayed transaction initiator, 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 Timeout value; otherwise, the transaction is discarded from the buffers.

8.6.4.2 Target Disconnect

The PCI 6520 returns a Target Disconnect to an initiator when the PCI 6520:

- Reaches an internal Address boundary
- Reaches a 4-KB boundary for a Posted Memory Write cycle
- Cannot accept further Write data
- Contains no further Read data to deliver

8.6.4.3 Target Abort

The PCI 6520 returns a Target Abort to an initiator when the PCI 6520:

- Returns a Target Abort from the intended target
- Detects a Master Abort on the target, and the Master Abort Mode bit is set (BCNTRL[5]=1; PCI:3Eh)
- Cannot obtain Delayed Read data from the target nor deliver Delayed Write data to the target after 2^{24} attempts

When returning a Target Abort to the initiator, the PCI 6520 sets the Status register Signaled Target Abort bit corresponding to the initiator interface (PCISR[12 or 11]=1; PCI:06h).

9 PCI-X BUS OPERATION

This section describes PCI-X transactions that the PCI 6520 responds to as a target and those it initiates as a master when operating with one or both of its interfaces in PCI-X mode.

9.1 OVERVIEW

Because the PCI 6520 is a PCI-X bridge (capable of operating in PCI-X mode), each of the PCI 6520 interfaces (primary or secondary) is capable of operating in Conventional PCI mode when a Conventional PCI device is installed (PCI-XCAP is connected to ground).

The PCI 6520 supports PCI-X mode in the following way:

- Source bridge for the primary bus informs the PCI 6520 of the primary bus mode, with the PCI-X initialization pattern at the rising edge of P_RSTIN#
- Senses S_XCAP_IN state and properly initializes the secondary bus devices
- Supports 64-bit addressing on both interfaces
- Implements a 64-bit AD Bus on either interface
- Completes all DWORD and Burst Memory Read transactions as Split transactions, if the transactions cross the PCI 6520 and the originating interface is in PCI-X mode
- PCI 6520 does not support system topologies, Special Cycle, and Interrupt Acknowledge cases (in Conventional PCI nor PCI-X mode)

9.2 GENERAL BUS RULES

When operating in PCI-X mode, the PCI 6520 adheres to the following PCI-X transactions rules:

- Address phase is the phase in which FRAME# is asserted. In the Address phase, the AD Bus contains the starting address (*except* for Split Completions) and the CBE Bus contains the command.
- Starting address of Configuration Read and Write transactions is aligned to a DWORD boundary. Split Completion transactions use only a partial starting address.
- Attribute phase follows the Address phase(s).
- In the Target Response phase, the CBE Bus is ***reserved*** and driven high (the clock after the Attribute phase).
- Burst transactions include the Byte Count (number of bytes between the first byte of the transaction and the last byte of the Sequence, inclusive) in the attributes.
- DWORD transactions do ***not*** use a Byte Count and ***reserve*** the Attribute Byte Count field.
- Target Response phase is one or more clocks after the Attribute phase and ends when the target asserts DEVSEL#. If no target asserts DEVSEL#, there are no Data phases (Master Abort).
- Transactions using the I/O Read and Write and Configuration Read and Write commands (one Data phase transaction) are initiated only as 32-bit transactions (REQ64# is de-asserted, as in Conventional PCI mode).
- For Memory Write transactions, the Byte Count is not adjusted for bytes whose Byte Enables are de-asserted within the transaction (Byte Count is the same).

Table 9-1 lists the definitions for typical Conventional PCI (provided for reference) and PCI-X transaction phases.

Table 9-1. Transaction Phase Definitions

Conventional PCI Transaction Phases	Description	PCI-X Transaction Phases	Description
Address	One clock for SAC. Two clocks for DAC.	Address	One clock for SAC. Two clocks for DAC.
Data	Clocks (after the Address phase) in which Data transfer is allowed.	Attribute	One clock. Provides further details about the transaction.
Initiator Termination	Initiator signals the end of the transaction on the last Data phase.	Target Response	One or more clocks when the target claims the transaction by asserting DEVSEL#.
Turn-Around	Idle clock for changing from one signal driver to another.	Data	Clocks in which Data transfer is allowed.
		Initiator Termination	Initiator signals the end of the transaction one clock before the last Data phase.
		Turn-Around	Idle clock for changing from one signal driver to another.

9.2.1 Initiator Rules

When operating in PCI-X mode, the PCI 6520 adheres to the following rules when initiating a transaction:

- Begins a transaction by asserting FRAME# within two clocks after GNT# is asserted and the bus is idle (within six clocks if the transaction uses a Configuration command).
- Asserts IRDY# two clocks after the Attribute phase (in PCI-X, wait states are not allowed for initiators).
- De-asserts FRAME# one clock before the last Data phase (for four or more Data phases) or two clocks after the target asserts TRDY# for a single Data phase (or terminates the transaction with another method).
- De-asserts IRDY# one clock after the last Data phase (for four or more Data phases) or two clocks after the target asserts TRDY# for a single Data phase (or terminates the transaction with another method).
- Ends the transaction as a Master Abort if no target asserts DEVSEL# on or before the SUB decode time.
- For Write and Split Completion transactions, the PCI 6520 drives the first data value on the AD Bus two clocks after the Attribute phase and advances to the second data value (for Burst transactions with more than one Data phase) two clocks after the target asserts DEVSEL# (in anticipation of the target asserting TRDY#). If the target inserts wait states (by not asserting TRDY# after DEVSEL# assertion), the PCI 6520 toggles between the first and second data values until the target asserts TRDY# (or terminates the transaction with another method).
- Terminates a transaction when the Byte Count is satisfied. Disconnects a Burst transaction (before the Byte Count is satisfied) only on an ADB.
- PCI 6520 retains 64 clocks of Latency Timer value, and Disconnects the current transaction on the next ADB if the Latency Timer expires and GNT# is de-asserted.

9.2.2 Target Rules

When operating in PCI-X mode, the PCI 6520 adheres to the following rules when responding to a transaction as a target:

- Memory address range for the PCI 6520 (and all devices) is no smaller than 128 bytes and aligned to ADBs.
- Claims the transaction by asserting DEVSEL#, using Decode B.
- Does not signal Wait State after the first Data phase. Signals Split Response, Target Abort, or Retry within eight clocks of FRAME# assertion and signals Single Data Phase Disconnect (on the first Data phase only), Data Transfer, or Disconnect at the next ADB within 16 clocks of FRAME# assertion.
- May signal Target Abort on any Data phase, regardless of its relationship to an ADB.
- If DEVSEL#, STOP#, and TRDY# are not de-asserted, the PCI 6520 de-asserts these signals one clock after the last Data phase, then floats them one clock after that.

9.2.3 Bus Arbitration Rules

The following are the bus arbitration rules:

- PCI 6520 asserts and de-asserts REQ# on any clock (no requirement to de-assert REQ# after a Target Termination).
- PCI 6520 de-asserts REQ# on any clock, independent of whether GNT*x*# is asserted (without transaction initiation after GNT*x*# assertion).
- Arbiter asserts GNT*x*# on any clock if all the GNT*x*# signals are de-asserted. GNT*x*# remain asserted for a minimum of five clocks while the bus is idle or until the initiator asserts FRAME# or de-asserts REQ#.
- If only one device asserts REQ#, the Arbiter holds GNT*x*# asserted to that device.
- If the Arbiter de-asserts GNT*x*# to one device, it must wait until the next clock to assert GNT*x*# to another device.
- Fast Back-to-Back transactions are ***not*** allowed in PCI-X mode.

9.2.4 Configuration Transaction Rules

When operating in PCI-X mode:

- Initiators must drive the address for four clocks before asserting FRAME#
- Transaction must include the target Device Number in AD[15:11] of the Address phase
- Type 0 Configuration Write transaction target stores its Device and Bus Numbers in its internal registers

9.2.5 Parity Error Rules

- In PCI-X mode, when the PCI 6520 detects a Data Parity error while receiving data (*for example*, the target of a write or Split Completion, or initiator of a read), the PCI 6520 asserts PERR# on the second clock after PAR and PAR64 are driven (one clock later than Conventional PCI mode).
- During Read transactions, as a target, the PCI 6520 drives PAR and PAR64 on clock *n+1* for the Read data it drives on clock *n* and the Byte Enables driven by the initiator on clock *n+1*.
- During Write transactions, as an initiator, the PCI 6520 drives PAR and PAR64 on clock *n+1* for the Write data and Byte Enables it drives on clock *n*.
- PCI 6520 asserts SERR# when it detects a Parity error during an Attribute phase.

9.2.6 Bus Data Width Rules

- In PCI-X mode, the bus data width of each transaction is determined with a handshake protocol (similar to Conventional PCI protocol) on ACK64# and REQ64#
- PCI 6520 implements 64-bit in both its interfaces and is capable of generating a 64-bit Memory address
- Attribute phase is always a single clock long for both 64- and 32-bit initiators
- Only Burst transactions (Memory commands other than Memory Read DWORD) use 64-bit Data transfers

9.2.7 Split Transaction Rules

- Transactions that are terminated with Split Response result in one or more Split Completion transactions
- Split Completions contain Read data or a Split Completion Message, but not both
- If returning Read data, the completer must return all data (the full Byte Count) unless an error occurs
- Requester must accept all Split Completion Data phases, and can terminate a Split Completion with Data Transfer or Target Abort
- If the request is a Write transaction, or if the completer encounters an error while executing the request, the completer sends a Split Completion Message to the requester

9.3 PCI-X SEQUENCES

One or more transactions of a single logical transfer is called a *Sequence*. If a Sequence is broken into more than one transaction, the bytes of all these transactions are included in the request Byte Count that started the Sequence. Each transaction in the same Sequence carries the same Requester ID and Tag.

The PCI 6520 does not initiate a new Sequence using the same Tag until the previous Sequence using that Tag is complete.

The Sequence has more than one transaction if it is a Burst write and Disconnected by the initiator or target. The initiator must resume the Sequence by initiating another Burst Write transaction, using the same command and adjusting the starting address and data Byte Count.

- If the Sequence is a Burst write and the target signals a Target Abort or no target responds (Master Abort), the Sequence ends when the transaction terminates.
- If the Sequence is a Burst read that executes as an Immediate Transaction, the Sequence terminates when the transaction terminates. The requester is not required to resume an immediate read Sequence.

When a Sequence crosses the PCI 6520, the number of transactions within the Sequence on each bus is determined by the device behavior on each bus. The PCI 6520 maintains Split Completions, in address order, for the same Sequence.

9.4 ADB AND BUFFER SIZE

An Allowable Disconnect Boundary (ADB) is an aligned 128-byte address [the lower seven bits are zeros (0)]. ADBs are the same, regardless of transaction width. To proceed from one ADB to the next, a Burst transaction requires 16 Data phases on a 64-bit bus and 32 Data phases on a 32-bit bus.

After a Burst Data transfer starts and the target accepts more than a single Data phase, the transaction can only be stopped using one of the following methods:

- Target or Initiator Disconnect at an ADB
- Transaction Byte Count is satisfied
- Target Abort

If a Burst transaction is Disconnected (by the initiator or target) on an ADB, the last byte address transferred is 7Fh (modulo 80h).

9.5 DEPENDENCIES BETWEEN AD AND CBE BUSES

PCI-X Memory, I/O, and Configuration spaces are addressable at the byte level as Conventional PCI mode. Bytes appear on byte lanes according to their individual Byte address and transfer width, as listed in Table 9-2.

For Burst transactions, the PCI-X requester uses the full address bus to indicate the starting Byte address (including AD[1:0]) and includes the Attribute Byte Count field.

For I/O and Memory DWORD transactions, the PCI-X initiator uses the full Address bus to indicate the starting address (including AD[1:0]). The Attribute Byte Count field is ***reserved***. The ending address is always the last byte of the Dword.

Note: *Interrupt Acknowledge and Special Cycle transactions have no address.*

The CBE Bus is ***reserved*** and driven high throughout Burst transactions (*except* Memory Writes). During the Data phases of all DWORD and Memory Write transactions, the CBE Bus is used as Byte Enables.

Table 9-2. Byte Lane Assignments

	Byte Lane Transfer Width	
Byte Address AD[2:0]	**32-Bit**	**64-Bit**
000b	0	0
001b	1	1
010b	2	2
011b	3	3
100b	0	4
101b	1	5
110b	2	6
111b	3	7

9.6 PCI-X COMMAND ENCODING

PCI-X command codes are listed in Table 9-3. The initiator must not generate ***reserved*** commands and targets should ignore transactions using ***reserved*** commands.

For all commands (*except* DAC), the transaction command appears on CBE[3:0]# during the single Address phase. For transaction with DACs, CBE[3:0]# contains the DAC command in the first Address phase and the Transaction command in the second Address phase.

The Alias to Memory Read and Write Block commands are ***not*** generated by the initiators, and are treated as Memory Read Block and Memory Write Block, respectively, by targets.

For 64-bit transactions, CBE[7:4]# contain the transaction command in both Address phases.

Table 9-3. PCI-X Command Encoding

CBE[3:0]#	PCI-X Command	Length	Byte Enable Usage
0000b	Interrupt Acknowledge ***(Not Supported)***	Dword	Valid
0001b	Special Cycle ***(Not Supported)***	Dword	Valid
0010b	I/O Read	Dword	Valid
0011b	I/O Write	Dword	Valid
0100b	***Reserved***	N/A	N/A
0101b	***Reserved***	N/A	N/A
0110b	Memory Read DWORD	Dword	Valid
0111b	Memory Write	Burst	Valid
1000b	Alias to Memory Read Block	Burst	***Reserved*** / Driven High
1001b	Alias to Memory Write Block	Burst	***Reserved*** / Driven High
1010b	Configuration Read	Dword	Valid
1011b	Configuration Write	Dword	Valid
1100b	Split Completion	Burst	***Reserved*** / Driven High
1101b	DAC	N/A	N/A
1110b	Memory Read Block	Burst	***Reserved*** / Driven High
1111b	Memory Write Block	Burst	***Reserved*** / Driven High

9.7 ATTRIBUTES

The transaction is further defined by additional information included in the Attribute phase (always single clock, regardless of the address or data transfer width). The upper buses (AD[63:32] and CBE[7:4]#) of 64-bit devices are ***reserved*** and driven high during the Attribute phase. PCI-X transactions have three attribute formats. The Requester Attribute format applies to all transactions, *except* Type 0 Configuration Read and Write transactions and Split Completion transactions. Figure 9-1 illustrates the Requester Attribute bit assignments.

The Requester ID (AD[23:8]) is the combination of the Requester Bus, Device, and Function Numbers.

The Sequence ID (AD[28:8]) is the combination of the Requester ID and Tag. Table 9-4 lists the Requester Attribute bit definitions.

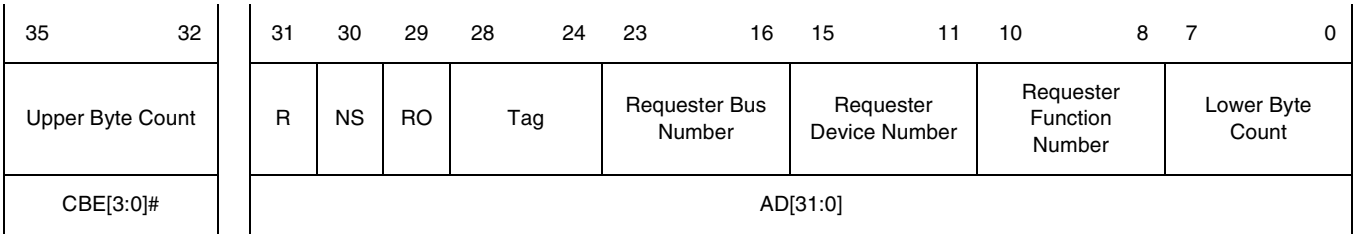


Figure 9-1. Requester Attribute Bit Assignments

Table 9-4. Requester Attribute Bit Definitions

Bit(s)	Attribute	Function
CBE[3:0]# AD[7:0]	Upper Byte Count Lower Byte Count *Example*: **Byte Count[11:0] Bytes** 0000_0000_0001b 1 0010_0101_0110b 598 1111_1111_1111b 4095 0000_0000_0000b 4096	Indicates the number of bytes the initiator plans to move in the remainder of the Sequence. If the transaction is Disconnected for any reason and the initiator continues the Sequence, the initiator must adjust the Byte Count contents in the subsequent transactions of the same Sequence to be the number of bytes remaining in the Sequence.
AD[10:8]	Requester Function Number	Contains the requester Function Number (the function in the Configuration address to which the function responds).
AD[15:11]	Requester Device Number	Contains the Device Number assigned to the requester (from the PCI-X Bridge Status register Device Number bits, PCIXBSR[7:3]; PCI:F4h).
AD[23:16]	Requester Bus Number	Identifies the requester Bus Number (from the PCI-X Bridge Status register Bus Number bits, PCIXBSR[15:8]; PCI:F4h).
AD[28:24]	Tag	Uniquely identifies up to 32 Sequences from a single initiator. The initiator assigns a unique Tag to each Sequence that begins before the previous one ends. Tag identifies the appropriate Split Completion transaction.
AD29	Relaxed Ordering Attribute (RO)	Devices are allowed to set AD29 to load Sequences and must clear it for Control and Status Sequences. Split Read requesters are unaffected by this bit.
AD30	No Snoop (NS)	If an initiator sets AD30, the initiator guarantees that the locations between the starting and ending address, inclusive, of this Sequence are not stored in a system cache. • If a Write transaction is Disconnected, the initiator must not change this attribute on a subsequent transaction in the same Sequence • If an immediate Read transaction Disconnects, the Sequence ends PCI 6520 ignores AD30 and forwards it, unmodified, with the transaction.
AD31	***Reserved*** (R)	Set to 0 by the requester and ignored by the completer.

9.8 BURST TRANSACTIONS

A burst transaction is a transaction that uses one of the following commands:

- **Burst Write transactions**—Memory Write, Memory Write Block, or Alias to Memory Write Block
- **Burst Read transactions**—Memory Read Block or Alias to Memory Read Block commands
- Split Completion

Burst transactions (may be initiated as 64- or 32-bit) transfer data on one or more Data phase, up to that required to satisfy the maximum Byte Count. Burst transactions use the full Address bus (including AD[2:0]) to specify the transaction starting Byte address. Burst transactions can start on one side and end on the other of an ADB, a Memory Page boundary, the first 4 GB Memory Address boundary and so forth.

During Burst transaction Data phases, the CBE Bus is ***reserved*** and driven high by the initiator for all transactions (*except* Memory Writes).

9.8.1 Burst Write and Split Completion

The PCI 6520 executes Burst Write and Split Completion transactions as Immediate transactions, as follows:

- As an initiator, the PCI 6520 is the command source, address, attribute, and data.
- As a target, terminates a Burst Write transaction with Target Abort, Single Data Phase Disconnect, Wait State, Data Transfer, Retry, or Disconnect at the next ADB.
- PCI 6520 terminates a Split Completion transaction with Target Abort, Wait State, or Data Transfer, Retry, or Disconnect at the next ADB. Targets are not allowed to respond with Split Response.
- CBE Bus is ***reserved*** and driven high after the Address phase of all Burst writes and Split Completion transactions (*except* Memory Writes). All bytes between the starting and ending address, inclusive, are included in these transactions.
- If the target inserts initial wait states, it must do so in pairs of clocks, and the initiator must toggle between the first and second data patterns until the target begins accepting data.

9.8.2 Burst Read Transactions

Burst Read transactions use the Memory Read Block or Alias to Memory Read Block commands.

As an initiator, the PCI 6520 is the source of the command, address, and attributes (the completer is the source of the data).

- As a target, the PCI 6520 responds to a Burst Read transaction with Split Response, Target Abort, Single Data Phase Disconnect, Wait State, Data Transfer, Retry, or Disconnect at the next ADB.
- If the PCI 6520 responds (as a target) by signaling Single Data Phase Disconnect, Data Transfer, or Disconnect at the next ADB, Data transfers during the Read transaction.
- If the PCI 6520 responds (as a target) with a Split Response, no Data transfers during the Read transaction but is later transferred in one or more Split Completion transactions.

9.9 DWORD TRANSACTIONS

A PCI-X DWORD transaction uses Interrupt Acknowledge, Special Cycle, I/O Read, I/O Write, Configuration Read, Configuration Write, or Memory Read DWORD. The PCI 6520 considers the following PCI-X rules:

- DWORD transactions must be initiated as 32-bit transfers (REQ64# must be de-asserted and AD[63:32], CBE[7:4]#, and PAR64 are not used)
- DWORD transactions retain a single Data phase and affect more than a single Dword
- ***Reserved*** attributes must be set to 0 by the initiator and ignored by the target
- Byte Enables are not valid until two clocks after the Attribute phase, and PCI-X completer with locations that exhibit read side effects must not start a Read operation until the Byte Enables are valid
- Interrupt Acknowledge has no address, and drives any value on the AD[31:0] bus during the Address phase

9.10 DEVICE SELECT TIMING

A PCI-X target claims transactions by asserting DEVSEL#, using one of the timings listed in Table 9-5 (Conventional PCI mode DEVSEL# timing is listed for reference).

Table 9-5. DEVSEL# Timing

Decode Speed after Address Phase[1]	Conventional PCI	PCI-X
1 Clock	Fast	N/A
2 Clocks	Medium	Decode A
3 Clocks	Slow	Decode B
4 Clocks	SUB	Decode C
5 Clocks	N/A	N/A
6 Clocks	N/A	SUB[2]

1. Decode speeds are measured from the second Address phase in the case of DAC.

2. If no target asserts DEVSEL# within the SUB decode time, the initiator ends the transaction as a Master Abort.

After asserting DEVSEL#, a target must complete the transaction with one or more Data phases by signaling Split Response, Target Abort, Single Data Phase Disconnect, Wait State, Data transfer, Retry, or Disconnect at the next ADB. The PCI 6520 uses Decode B timing.

9.11 WAIT STATES AND TARGET INITIAL LATENCY

When initiating a transaction, the PCI 6520 (and all PCI-X initiators) does not insert wait states. The PCI 6520:

- Asserts IRDY# two clocks after the Attribute phase and drives Write data on the AD Bus or prepares to accept Read data
- After IRDY# is asserted, it continues to assert IRDY# until the transaction ends
- When responding to a transaction, it (and all PCI-X targets) inserts wait states
- Inserts wait states (in pairs of clock for Burst write and Split Completion transactions) only on the initial Data phase
- For transactions of more than one Data phase, it does not signal a wait state

The target initial latency is measured from the clock in which the initiator asserts FRAME#, to the clock in which the target signals something other than a wait state. Table 9-6 delineates the target initial latency for all DEVSEL# timing combinations, number of Address phases, and numbers of wait states.

The maximum number of initial wait states a PCI-X target is allowed to insert depends upon how the target terminates the transaction.

- Target must signal Split Response, Target Abort, or Retry within eight clocks of FRAME# assertion
- Target must signal Single Data Phase Disconnect, Data Transfer, or Disconnect at the next ADB within 16 clocks of FRAME# assertion
- If large numbers of wait states are required, executing the transaction as a Split Transaction provides more efficient bus use
- On a Read transaction, the target is allowed to insert any number of initial wait states up (need not be in pairs for Read transactions) to the maximum number

Table 9-6. Target Initial Latency

Wait States	SAC				DAC			
	DEVSEL# Timing				DEVSEL# Timing			
	A	B	C	SUB	A	B	C	SUB
0	3	4	5	7	4	5	6	8
1	4	5	6	8	5	6	7	9
2	5	6	7	9	6	7	8	10
3	6	7	8	10	7	8	9	11
4	7	8	9	11	8	9	10	12
5	8	9	10	12	9	10	11	13
6	9	10	11	13	10	11	12	14
7	10	11	12	14	11	12	13	15
8	11	12	13	15	12	13	14	16
9	12	13	14	16	13	14	15	N/A
10	13	14	15	N/A	14	15	16	N/A
11	14	15	16	N/A	15	16	N/A	N/A
12	15	16	N/A	N/A	16	N/A	N/A	N/A
13	16	N/A	N/A	N/A	N/A	N/A	N/A	N/A

9.12 SPLIT TRANSACTIONS

Targets must not assume that the PCI 6520 (as an initiator) repeats a transaction terminated with Retry. Split Transactions improve bus efficiency for transactions accessing targets that exhibit long latency. Split Transactions (in PCI-X mode) replace Delayed Transactions (in Conventional PCI mode). A Split Transaction consists of at least two separate bus transactions—a requester-initiated Split Request, and one or more completer-initiated Split Completions.

The following is the sequence of events that occurs for Non-Memory Write transactions in which the initiator and target reside on opposite sides of the PCI 6520 bridge. These transactions are Memory Read Block, Alias to Memory Read Block, Memory Read DWORD, I/O Read, I/O Write, Configuration Read, or Configuration Write command.

1. Split Request (transaction terminated with a Split Response) occurs.
2. After signaling a Split Response, the PCI 6520 forwards the transaction to the completer side.
3. Completer terminates the transaction after Immediate Completion (Abort) or Split Completion cycle.
4. PCI 6520 initiates a Split Completion transaction to send the received Read data or a Split Completion Message to the requester.
5. PCI 6520 becomes the initiator of the Split Completion transaction, and the requester becomes the target (the requester and PCI 6520 switch roles).

   Split Completions for the same Sequence must be initiated in address order.

When the PCI 6520 terminates a Read transaction with Split Response, the PCI 6520 transfers the entire requested Byte Count as a Split Completion. The PCI 6520 provides four ADQs of Buffer space for the Split Completion.

9.12.1 Split Completion Transaction

A Split Completion transaction is a Burst transaction (includes the Byte Count in the Attribute phase) that uses the Split Completion command.

The following are the characteristics of Split Completion cycles:

- CBE Bus is ***reserved*** and driven high during all Data phases
- Split Completion transactions can be initiated as 64- or 32-bit transfers
- Target of a Split Completion is the requester that initiated the Split Request
- Completer stores the Requester ID (Bus, Device, and Function Numbers) from the Split Request Attribute phase
- Requester ID becomes part of the Split Completion address driven on the AD Bus during the Split Completion Address phase
- Requester uses the Requester ID to recognize Split Completions that correspond to its Split Request
- Split Completion attributes carry information about the completer rather than the requester
- Completer adjusts the address and Byte Count (to the portion remaining in the Sequence) each time a Split Completion resumes
- Completer can Disconnect a Split Completion transaction only on an ADB
- Completer must maintain the Split Completion data in address order
- If the Split Request is a DWORD transaction, the Lower Address bits in the Split Completion address are set to 0h and the Byte Count bit in the completer attributes is set to 4h
- If the Split Request is a Write transaction, a Split Completion Message is driven on AD[31:0], regardless of the Byte Enables asserted in the Split Request
- If the Split Request is a Read transaction, data is driven on AD[31:0] during the Split Completion Data phase (only byte lanes corresponding to the Byte Enables in the Split Request contain valid data)

9.12.2 Immediate Completion by the Completer

When the PCI 6520 forwards a Split Request, it expects that the completer immediately completes the transaction (*for example*, executes the transaction as an Immediate Transaction rather than a Split Transaction) or for the transaction to end with a Master Abort.

If the completer immediately completes the transaction, the PCI 6520 creates a Split Completion transaction (Split Completion address and completer attributes) to return to the requester.

Split Completions are created in the following way:

- PCI 6520 creates the Split Completion address from the original request in the same way as a completer
- For the Completer Attributes, the PCI 6520 creates the Completer ID for the bus in which the Immediate Completion occurred
- If the Immediate Completion occurred on the primary bus, the PCI 6520 supplies the Bus, Device, and Function Numbers from the PCI-X Bridge Status register (PCIXBSR[15:0]; PCI:F4h)
- If the Immediate Completion occurred on the secondary bus, the PCI 6520 supplies the Bus Number from the Secondary Bus Number register (PCISBNO; PCI:19h) and sets the Device and Function Number bits to 0 (PCIXBSR[7:0]=0h; PCI:F4h)

If the Split Request is a Write transaction and the completer immediately completes it, the PCI 6520 also creates a Split Completion Message for the Split Completion Data phase.

9.12.3 Split Completion Address

The Split Completion address is driven on the AD Bus during the Address phase of Split Completion transactions. (Refer to Figure 9-2.) The completer copies all information from the Split Request Address and Attribute phases. Table 9-7 lists the Split Completion Address bit definitions.

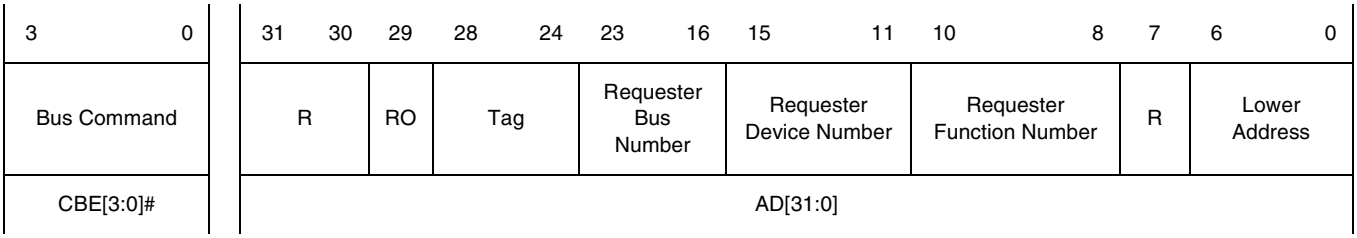


Figure 9-2. Split Completion Address

Table 9-7. Split Completion Address Bit Definitions

Bit(s)	Attribute	Function
AD[6:0]	Lower Address	PCI 6520 uses this information when forwarding a Split Completion to another bus operating in PCI-X mode, to determine where the Split Completion starts relative to an ADB. Completer copies AD[6:0] from the least significant seven bits of the Split Request address, regardless of the command used by the Split Request, if: • Split Request for this Sequence was a Burst read • First Split Completion of the Sequence • Split Completion is not a Split Completion Message AD[6:0] are set to 0h when the Sequence resumes if: • Split Completion is Disconnected on an ADB • Split Request was a DWORD transaction • Split Request was a Split Completion Message
AD7	***Reserved*** (R)	Set to 0 by the target initiator.
AD[10:8]	Requester Function Number	Completer copies AD[10:8] from the corresponding bits of the requester attributes. The requester uses this information to identify the appropriate Split Completions.
AD[15:11]	Requester Device Number	Completer copies AD[15:11] from the corresponding bits of the requester attributes. The requester uses this information to identify the appropriate Split Completions.
AD[23:16]	Requester Bus Number	Completer copies AD[23:16] from the corresponding bits of the requester attributes. The requester uses this information to identify the appropriate Split Completions. PCI 6520 uses this field to identify transactions to forward.
AD[28:24]	Tag	Completer copies AD[28:24] from the corresponding bits of the requester attributes. The requester uses this information to identify the appropriate Split Completions.
AD29	Relaxed Ordering Attribute (RO)	Completer optionally copies AD29 from the corresponding bit of the Requester Attributes Bridges throughout the system, using this bit to influence transaction ordering.
AD[31:30]	***Reserved*** (R)	Set to 00b by the requester and ignored by the completer.
CBE[3:0]#	Bus Command	Contains the Split Completion command.

9.12.4 Completer Attributes

The Attribute phase of a Split Completion contains the completer attributes (a combination of the Completer ID and information about the Sequence stored from the Split Request). (Refer to Figure 9-3 and Table 9-7.)

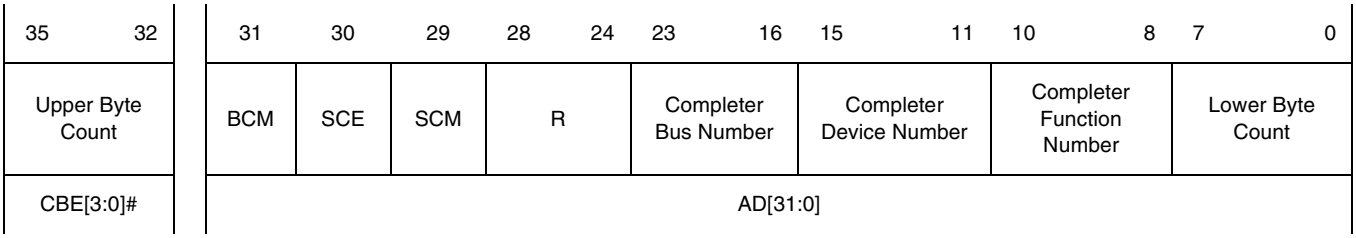


Figure 9-3. Completer Attribute Bit Assignments

Table 9-8. Completer Attribute Bit Definitions

Bit(s)	Attribute	Function
CBE[3:0]# AD[7:0]	Upper Byte Count Lower Byte Count *Example*: **Byte Count[11:0] Bytes** 0000_0000_0001b 1 0010_0101_0110b 598 1111_1111_1111b 4095 0000_0000_0000b 4096	Indicates the number of bytes the initiator plans to move in this Split Completion. If the transaction is Disconnected for any reason and the initiator continues the Sequence, the initiator must adjust the Byte Count field contents in the subsequent transactions of the same Sequence to be the number of bytes remaining in this Sequence. If the Split Completion is a Split Completion Message, the completer sets the Byte Count to four (0000_0000_0100b).
AD[10:8]	Completer Function Number	Contains the Function Number (function in the Configuration address to which the function responds) of the completer within the device.
AD[15:11]	Completer Device Number	Contains the Device Number assigned to the completer (from the PCI-X Bridge Status register Device Number bits, PCIXBSR[7:3]; PCI:F4h).
AD[23:16]	Completer Bus Number	Identifies the requester's Bus Number (from the PCI-X Bridge Status register Bus Number bits, PCIXBSR[15:8]; PCI:F4h).
AD[28:24]	***Reserved*** (R)	Set to 0h by the requester and ignored by the completer.
AD29	Split Completion Message (SCM)	Set to 0 by the completer if the Split Completion contains Read data. Set to 1 by the completer if the Split Completion contains a Split Completion Message.
AD30	Split Completion Error (SCE)	Set to 0 by the completer if the transaction is a Split Completion Message that is an error message.
AD31	Byte Count Modified (BCM)	Set to 1 by the completer if the Byte Count bit for this Split Completion contains a number smaller than the full remaining transaction Byte Count (to Disconnect the Split Completion at the first ADB). Set to 0 by the completer if the Byte Count bit contains the full remaining Byte Count of the Split Request, or if the Split Completion is a Split Completion Message. ***Note:*** *The BCM bit is used only for Split Completions resulting from Burst Read transactions (Memory Read Block and Alias to Memory Read Block) and is set to 0 for Split Completions resulting from all other commands.*

9.12.5 Split Completion Acceptance Requirement

The PCI 6520 accepts all Split Completions resulting from its own Split Requests. The PCI 6520 asserts DEVSEL# on all Split Completions in which the Sequence ID (Requester ID and Tag) corresponds to a PCI 6520-issued Split Request.

If the Split Completion Requester ID is corrupt, it is possible that the ID matches that of an actual system device. The PCI 6520 does ***not*** assert DEVSEL# (*for example*, ignores the corrupt transaction) if the Requester ID matches that of the PCI 6520, but the Tag does not match any outstanding requests from the PCI 6520.

When the PCI 6520 asserts DEVSEL# for a Split Completion, it accepts the entire Byte Count requested without signaling Split Response or Single Data Phase Disconnect.

If the PCI 6520 is executing one Split Transaction from one interface (issued Split Response on that interface), the PCI 6520 terminates with Retry Unposted transactions on that interface until the previous Split Transaction completes.

9.12.6 Split Completion Messages

Split Completion transactions include a message if the Completer Attributes SCM bit is set (AD29=1). A Split Completion Message is a single DWORD Burst transaction. The Remaining Lower Address bits in the Split Completion address are set to 0 and the Completer Attributes Byte Count is set to 4h for all Split Completion Messages. (Refer to Table 9-9.) The CBE Bus is ***reserved*** and driven high during the Data phase of a Split Completion Message.

The following illustrates the functions of Split Completion Messages:

- Notifies the requester when a Split Write Request (I/O or Configuration) has completed.
- Indicates error conditions in which delivery of data for a Read request or execution of a Write request is not possible.
- Terminates a Sequence, regardless of the amount of bytes remaining to be sent.
- Split Completion Message Byte Count bit indicates the number of bytes that were not sent for this Sequence (if the Split Request was a Burst read). The remaining Lower Address bits indicate the lower seven bits of the starting address of the remainder of the Sequence. (Refer to Figure 9-4, Table 9-9 and Table 9-10 for further details.)

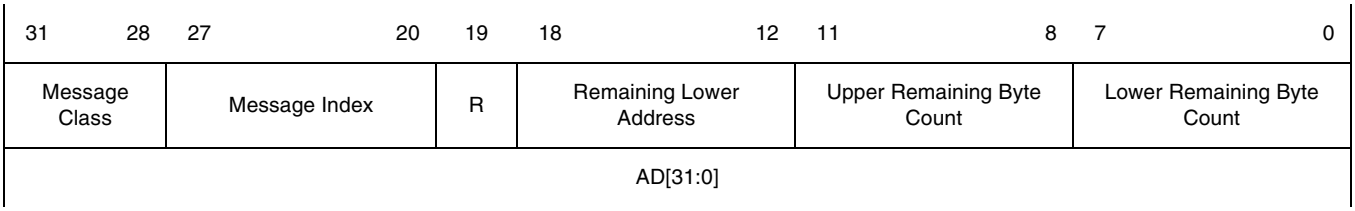


Figure 9-4. Split Completion Message Attribute Bit Assignments

Table 9-9. Split Completion Message Bit Definitions

Bit(s)	Attribute	Function
AD[11:0]	Upper and Lower Remaining Byte Count	If the Split Request was a Burst Memory Read, the completer sets this field to the number of bytes of Read data that were not sent. If the Split Request was a DWORD transaction, the completer sets AD[11:0] to 4h.
AD[18:12]	Remaining Lower Address	If the Split Request was a Burst Memory Read, this field contains the least significant seven bits of the address of the first byte of Read data that was not sent. If the Split Request was a DWORD transaction, the completer sets AD[18:12] to 0h.
AD19	***Reserved*** (R)	Set to 0 by the requester and ignored by the completer.
AD[27:20]	Message Index	Identifies the type of message within the message class. (Refer to Table 9-10 for further details.)
AD[31:28]	Message Class	Split Completion Messages classes. (Refer to Table 9-10 for further details.) Class values: 0000b = Write Completion 0001b = PCI-X Bridge Error 0010b = Completer Error 0011b – 1111b = ***Reserved*** ***Note:*** *The Write Completion class is used for Split Write Completion Messages. The Remaining Lower Address bits are set to 0 (AD[18:12]=0h) and the Upper and Lower Remaining Byte Count bits are set to 4h (AD[11:0]=4h) in the Data phase of this Split Completion Message.*

Table 9-10. PCI 6520 Error Message Indices

Index AD[27:20]	Message Class 0 (AD[31:28] = 0000b)	Message Class 1 (AD[31:28] = 0010b)	Message Class 2 (AD[31:28] = 0010b)
00h	Normal Write Completion	**Master Abort.** PCI 6520 encountered a Master Abort on the destination bus.	**Byte Count out of Range.** Completer uses this message if the sum of the Split Request address and the Byte Count exceed the completer Address range (over device boundary). The completer initiates Split Completion transactions with Read data up to the device boundary.
01h	***Reserved***	**Target Abort.** PCI 6520 encountered a Target Abort on the destination bus.	**Split Write Data Parity Error.** Completer sends this message when a DWORD Write transaction is terminated with Split Response and a Data Parity error detected.
02h	***Reserved***	**Write Data Parity Error.** PCI 6520 encountered a Data Parity error on an Unposted Write transaction on the destination bus.	**Device-Specific Error.** Completer uses this message if it encounters an error that prevents execution of the Split Request, and the error is not indicated by one of the other error messages. The PCI 6520 encodes specific error or diagnostic information from the lower four bits of this index.
03h – FFh	***Reserved***		

9.13 PCI-X MODE TRANSACTION TERMINATION

The PCI 6520 terminates a transaction using the following methods:

- **Initiator Termination**—Byte Count Disconnection or satisfaction
- **Master Abort Termination**—Target Termination, Disconnection, or Retry; Split Response; or Target Abort Termination

9.13.1 PCI 6520 Initiator Termination

9.13.1.1 Byte Count Disconnection or Satisfaction

PCI 6520 Disconnection on an ADB (before the Byte Count is satisfied), and PCI 6520 Termination at the end of the Byte Count, appear the same on the bus.

- PCI 6520 Termination in less than four Data phases occurs only if the starting address, Byte Count, and bus data width are such that the transaction has fewer than four Data phases
- When the PCI 6520 intends to Disconnect a transaction on the first ADB, and the bus data width is such that the starting address is less than four Data phases from the ADB, the PCI 6520 adjusts the Byte Count to terminate on that ADB

9.13.1.2 PCI 6520 Master Abort Termination

As an initiator, the PCI 6520 de-asserts FRAME# and IRDY# eight clocks after the Address phase(s) if no target asserts DEVSEL# within six clocks after the Address phase(s).

9.13.2 PCI 6520 Target Termination

As a target, after the PCI 6520 asserts DEVSEL# in the Target Response phase, the PCI 6520 completes the transaction with one or more Data phases. On each clock after the target response phase, the PCI 6520 signals its intention with a combination of target control signals (DEVSEL#, STOP#, and TRDY#). A Data phase ends each time the PCI 6520 signals anything other than Wait State. Table 9-11 delineates the alternatives for Data phase signaling.

After the PCI 6520 signals a Data Transfer on one Data phase, the transaction continues until the Byte Count is satisfied or the initiator terminates the transaction. (When the PCI 6520 signals a Split Response or Retry, the transaction immediately terminates, although the Byte Count is not satisfied.)

The transaction immediately terminates if the PCI 6520 signals a Split Response or Retry. Following the transaction termination, the PCI 6520 de-asserts DEVSEL#, STOP#, and TRDY# one clock after the last Data phase (if these signals are not currently de-asserted).

Table 9-11. PCI 6520 Data Phase Signaling

DEVSEL#	STOP#	TRDY#	Data Phase Signaling	Data Transfer	Transaction Status
De-Asserted	De-Asserted	De-Asserted	No Response: Master Abort[1]	No	Terminates
De-Asserted	De-Asserted	Asserted	Split Response	Yes/No[2]	Terminates
De-Asserted	Asserted	De-Asserted	Target Abort	No	Terminates
De-Asserted	Asserted	Asserted	Single Data Phase Disconnect	Yes	Terminates
Asserted	De-Asserted	De-Asserted	Wait State[3]	No	Continues
Asserted	De-Asserted	Asserted	Data Transfer	Yes	Continues
Asserted	Asserted	De-Asserted	Retry	No	Terminates
Asserted	Asserted	Asserted	Disconnect at Next ADB	Yes	Continues to an ADB

1. PCI 6520 does not allow this after asserting DEVSEL#.

2. No Data Transfer on a Split Response for a Read transaction. The target latches data on a Split Response for a Write transaction. In both cases, the transaction is not complete until the requester receives the Split Completion.

3. Wait states are allowed only on the first Data phase.

9.13.2.1 PCI 6520 Disconnects at Next ADB

The PCI 6520 signals a transaction Disconnection at the next ADB by asserting TRDY#, STOP#, and DEVSEL# on any Data phase of the transaction. After signaling Disconnect at the next ADB, the PCI 6520 can signal only Target Abort on all subsequent Data phases until the transaction ends.

If the PCI 6520 signals Disconnect at the next ADB:

- Four or more Data phases before an ADB, the initiator Disconnects the transaction on that ADB
- After signaling a Data Transfer on the first Data phase, and the transaction starting address is less than four Data phases from an ADB, the transaction crosses that ADB and continues to the next ADB

9.13.2.2 PCI 6520 Retry Termination

The PCI 6520 indicates that it is temporarily unable to complete the transaction by signaling Retry (asserting STOP# and DEVSEL# and holding TRDY# de-asserted on the first Data phase) under the following conditions:

- Initialization time after rising edge of RSTIN# did not elapse
- Buffers for accepting Memory Write or Split Completion transactions are currently full with previous data

The PCI 6520 discards all state information related to a transaction terminated with Retry.

9.13.2.3 PCI 6520 Split Response Termination

The PCI 6520 signals a Split Response if it has enqueued the transaction as a Split Request by asserting TRDY#, de-asserting DEVSEL#, and holding STOP# de-asserted on the first Data phase of the transaction. The PCI 6520 drives all AD Bus bits high during the clock in which it signals a Split Response of a Read transaction.

9.14 PCI-X MODE BUS AND DATA TRANSFER WIDTH

The PCI 6520 determines the bus data width to which it is attached by the REQ64# state at the rising edge of RSTIN#. PCI-X devices are allowed to implement a 64- or 32-bit interface. Addresses are driven in one or two clocks, depending on whether the transaction uses a memory command and the address is below the first 4-GB boundary. Attributes are always driven in a single clock for 64- and 32-bit devices. As in Conventional PCI mode, the width of a PCI-X Data transfer is negotiated between the initiator and the target on each transaction, using REQ64# and ACK64#.

Attributes:

- PCI 6520 is capable of generating addresses greater than 4 GB (when initiating Memory transactions)
- All PCI 6520 Prefetchable Memory Range registers are 64-bit registers
- Split Completions always have a single Address phase for 64- and 32-bit initiators

When executing a transaction with a DAC for a 64-bit transaction, the PCI 6520 drives the entire address (lower address on AD[31:0] and upper address on AD[63:32]) and both commands (DAC on CBE[3:0]# and transaction command on CBE[7:4]#) during the initial Address phase. On the following clock, the PCI 6520 drives the upper address on AD[63:0] and the transaction command on CBE[7:0]#.

The following requirements are applied to all PCI-X devices:

- PCI-X devices support a Status bit, indicating whether they are a 64- or 32-bit device.
- Only Burst transactions use 64-bit transfers. DWORD transactions use 32-bit transfers.
- ADBs are unaffected by the Data transfer width. A 32-bit device has twice as many Data phases between two ADBs.
- AD2 is 0 or 1, depending on the transaction starting Byte address. If AD2 of the starting byte address is 1 (the starting address is in the upper 32 bits of the bus), the 64-bit initiator must drive the data on AD[63:0] and the Byte Enables on CBE[7:0]# of the first Data phase.
- AD[63:32] and CBE[7:4]# are driven high during the Transaction Attribute phase from a 64-bit initiator.
- If the target concurrently asserts ACK64# and DEVSEL# (a 64-bit target), and the target inserts wait states, the initiator must toggle between the first and second QWORD Data phases on AD[63:0] and Byte Enables on CBE[7:0]#. If the transaction starts on an odd Dword, that Dword and its Byte Enables must be copied to the lower half of the bus each time the first Data phase repeats.
- If the target does not assert ACK64# (and the target inserts wait states), the initiator must toggle between the first and second DWORD Data phases of the transaction on AD[31:0] and Byte Enables on CBE[3:0]#.

9.15 CONNECTING CONVENTIONAL PCI AND PCI-X INTERFACES

This section provides the PCI 6520 translation of commands and protocol between Conventional PCI and PCI-X interfaces.

9.15.1 Conventional PCI Requester, PCI-X Completer

Table 9-12 summarizes the command translation requirements from a Conventional PCI transaction to a PCI-X transaction.

When the PCI 6520 prefetches more than a single Dword, it uses the Memory Read Block command. If a Memory Read Block command is used, the Byte Count is controlled by the PCI 6520 prefetch algorithm. The PCI 6520 Buffer memory writes transactions from its Conventional PCI interface, and counts the number of bytes to be forwarded to the PCI-X interface.

When the PCI 6520 forwards a transaction other than a Memory Write transaction:

- PCI 6520 terminates the transaction on the originating bus with Retry, stored the address, commands, and so forth, and enqueues a delayed request
- After the PCI 6520 finishes the request on the destination bus, it enqueues a delayed completion
- PCI 6520 follows Delayed Transaction rules on the requester side (Conventional PCI Bus) and Split Transaction rules on the completer side (PCI-X Bus)

Table 9-12. Conventional PCI-to-PCI-X Command Translation

Conventional PCI Command	PCI-X Command
I/O Read	I/O Read
I/O Write	I/O Write
Configuration Read	Configuration Read
Configuration Write	Configuration Write
Memory Read	Memory Read DWORD or Memory Read Block
Memory Read Line	Memory Read Block
Memory Read Multiple	Memory Read Block
Memory Write	Memory Write or Memory Write Block
Memory Write and Invalidate	Memory Write Block

9.15.2 PCI-X Requester, Conventional PCI Completer

Table 9-13 summarizes the command translation requirements from a PCI-X transaction to a Conventional PCI transaction.

The PCI 6520 translates a PCI-X Memory Read Block command into one of three Conventional PCI Memory Read commands, based on the Byte Count and starting address. If the:

- Starting address and Byte Count are such that only one or fewer Dwords are being read, the Conventional PCI transaction uses the Memory Read command
- PCI-X transaction reads more than one Dword, but does not cross a Cache Line boundary, the Conventional PCI transaction uses the Memory Read Line commands
- PCI-X transaction crosses a Cache Line boundary, the Conventional PCI transaction uses the Memory Read Multiple command

If PCI-X Memory Write Block command (or Memory Write command) Byte Enables are set and the command starts and ends on a Cache Line boundary, the PCI 6520 translates the command to the Memory Write command on the Conventional PCI interface.

When forwarding a transaction other than a Memory Write, the PCI 6520:

- Terminates the transaction on the originating bus with Split Response. After executing the transaction on the Conventional PCI interface, the PCI 6520 creates the Split Completion to return to the PCI-X requester.
- Follows Split Transaction rules on the requester side (PCI-X Bus) and Delayed Transaction rules on the completer side (Conventional PCI Bus).

Table 9-13. PCI-X-to-Conventional PCI Command Translation

PCI-X Command	Conventional PCI Command
I/O Read	I/O Read
I/O Write	I/O Write
Configuration Read	Configuration Read
Configuration Write	Configuration Write
Memory Read DWORD	Memory Read
Memory Read Block	Memory Read, Memory Read Line, or Memory Read Multiple
Memory Write	Memory Write or Memory Write and Invalidate
Memory Write Block	Memory Write or Memory Write and Invalidate

10 ADDRESS DECODING

This section describes address decoding, including Address ranges, Memory address decoding, ISA mode, and VGA and private device support.

10.1 OVERVIEW

The PCI 6520 uses three Address ranges to control I/O and Memory Transaction forwarding across the bridge. These address ranges are defined by Base and Limit Address registers in Configuration space.

10.2 ADDRESS RANGES

The PCI 6520 uses the following Address ranges to determine which I/O and Memory transactions are forwarded from the primary-to-secondary PCI Bus, and from the secondary-to-primary PCI Bus:

- One 32-Bit I/O Address range
- One 32-Bit Memory-Mapped I/O (non-prefetchable memory) range
- One 64-Bit Prefetchable Memory Address range

Transaction addresses falling within these ranges are forwarded downstream from the primary-to-secondary PCI Bus. Transaction addresses falling outside these ranges are forwarded upstream from the secondary-to-primary PCI Bus.

10.2.1 I/O Address Decoding

The PCI 6520 uses the following mechanisms, defined in Configuration space, to specify the I/O Address space for downstream and upstream forwarding:

- I/O Base and Limit Address registers (Base—PCIIOBAR; PCI:1Ch and PCIIOBARU16; PCI:30h, Limit—PCIIOLMT; PCI:1Dh and PCIIOLMTU16; PCI:32h)
- ISA Enable bit (BCNTRL[2]; PCI:3Eh)
- VGA Enable bit (BCNTRL[3]; PCI:3Eh)
- VGA Palette Snoop Enable bit (PCICR[5]; PCI:04h)

To enable downstream I/O transaction forwarding, the Command register I/O Space Enable bit must be set (PCICR[0]=1; PCI:04h). If the I/O Space Enable bit is not set, I/O transactions initiated on the primary bus are ignored. To enable upstream I/O transaction forwarding, the Command register Master Enable bit must be set (PCICR[2]=1; PCI:04h). If the Master Enable bit is not set, the PCI 6520 ignores I/O and Memory transactions initiated on the secondary bus. Setting the Master Enable bit also allows upstream Memory transaction forwarding.

Caution: If any configuration state affecting I/O transaction forwarding is changed by a Configuration Write operation on the primary bus when there are ongoing I/O transactions on the secondary bus, the PCI 6520 response to the secondary bus I/O transactions is unpredictable. Configure the I/O Base and Limit Address registers, and ISA Enable, VGA Enable, and VGA Palette Snoop Enable bits before setting the I/O Space Enable and Master Enable bits, and subsequently change these registers only when the primary and secondary PCI Buses are idle.

10.2.1.1 I/O Base and Limit Address Registers

The PCI 6520 implements one set of I/O Base and Limit Address registers in Configuration space that define an I/O Address range for downstream forwarding. The PCI 6520 supports 32-bit I/O addressing, which allows I/O addresses downstream from the PCI 6520 to be mapped anywhere in a 4-GB I/O Address space.

I/O transactions with addresses that fall inside the I/O Base and Limit register-defined range are forwarded downstream from the primary-to-secondary PCI Bus. I/O transactions with addresses that fall outside this range are forwarded upstream from the secondary-to-primary PCI Bus. The I/O range can be disabled 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 disabled, all I/O transactions are forwarded upstream (no I/O transactions are forwarded downstream).

The I/O Base register consists of an 8-bit field (PCIIOBAR; PCI:1Ch) and a 16-bit field (PCIIOBARU16; PCI:30h). The upper four bits of the 8-bit field define bits [15:12] of the I/O Base address.

The lower four Read-Only bits are hardcoded to 0001b to indicate 32-bit I/O addressing support. Bits [11:0] of the Base address are assumed to be 0h, which naturally aligns the Base address to a 4-KB boundary with a minimum granularity of 4 KB. The 16 bits contained in the I/O Base Upper 16 Bits register (PCIIOBARU16; PCI:30h) define AD[31:16] of the I/O Base address. All 16 bits are read/write. After a primary bus or chip reset, the I/O Base address value is initialized to 0000_0001h.

The I/O Limit register consists of an 8-bit field (PCIIOLMT; PCI:1Dh) and a 16-bit field (PCIIOLMTU16; PCI:32h). The upper four bits of the 8-bit field define bits [15:12] of the I/O Limit address. The lower four Read-Only bits are hardcoded to 0001b to indicate 32-bit I/O addressing support. Bits [11:0] of the Limit address are assumed to be FFFh, which naturally aligns the Limit address to the top of a 4-KB I/O Address block. The 16 bits contained in the I/O Limit Upper 16 Bits register (PCIIOLMTU16; PCI:32h) define AD[31:16] of the I/O Limit address. All 16 bits are read/write. After a primary bus or chip reset, the I/O Limit address value is reset to 0000_ 0FFFh.

Note: *Write these registers with their appropriate values before setting the Command register Master or I/O Space Enable bit (PCICR[2 or 0]=1; PCI:04h).*

10.3 MEMORY ADDRESS DECODING

The PCI 6520 has three mechanisms for defining Memory Address ranges for Memory transaction forwarding:

- Memory-Mapped I/O Base and Limit Address registers (PCIMBAR; PCI:20h and PCIMLMT; PCI:22h, respectively)
- Prefetchable Memory Base and Limit Address registers (Base—PCIPMBAR; PCI:24h and PCIPMBARU32; PCI:28h, Limit—PCIPMLMT; PCI:26h and PCIPMLMTU32; PCI:2Ch)
- VGA mode (BCNTRL[3]=1; PCI:3Eh)

This subsection describes the first two mechanisms. VGA mode is described in Section 10.5.1.

To enable downstream Memory transaction forwarding, the Command register Memory Space Enable bit must be set (PCICR[1]=1; PCI:04h). To enable upstream Memory transaction forwarding, the Command register Master Enable bit must be set (PCICR[2]=1; PCI:04h). Setting the Master Enable bit also enables upstream I/O transaction forwarding.

Caution: If any configuration state affecting Memory transaction forwarding is changed by a Configuration Write operation on the primary bus when there are ongoing memory transactions on the secondary bus, response to the secondary bus Memory transactions is unpredictable. 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 Space Enable and Master Enable bits, and subsequently change these registers only when the primary and secondary PCI Buses are idle.

10.3.1 Memory-Mapped I/O Base and Limit Address Registers

Memory-mapped I/O is also referred to as Non-Prefetchable memory. The Memory-Mapped I/O Base and Limit Address registers define an Address range that the PCI 6520 uses to determine when to forward Memory commands. The PCI 6520 forwards a Memory transaction from the primary-to-secondary interface if the Transaction address falls within the Memory-Mapped I/O Address range. The PCI 6520 ignores Memory transactions initiated on the secondary interface that fall into this Address range. Transactions that fall outside this Address range are ignored on the primary interface and forwarded upstream from the secondary interface (provided that the transactions do not fall into the Prefetchable Memory range, or are not forwarded downstream by the VGA mechanism).

The Memory-Mapped I/O Address range supports only 32-bit addressing. *P-to-P Bridge r1.2* 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 and a maximum range of 4 GB.

The Memory-Mapped I/O Address range is defined by a 16-bit Memory-Mapped I/O Base Address register (BAR) and a 16-bit Memory-Mapped I/O Limit Address register (PCIMBAR; PCI:20h and PCIMLMT; PCI:22h, respectively). The upper 12 bits of each of these registers correspond to bits [31:20] of the Memory address. The lower four bits are hardcoded to 0h. The lower 20 bits of the Memory-Mapped I/O Base address are assumed to be 0h, which results in a natural alignment to a 1-MB boundary. The lower 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.

Note: *Write the PCIMBAR and PCIMLMT registers with their appropriate values before setting the Command register Memory Space Enable or Master Enable bit.*

To disable 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.

10.3.2 Prefetchable Memory Base and Limit Address Registers

Locations accessed in the Prefetchable Memory Address range must have true memory-like behavior and not exhibit side effects when read (*that is*, extra reads to a prefetchable memory location must ***not*** have side effects). The PCI 6520 prefetches for all types of Memory Read commands in this Address space.

The PCI 6520 Prefetchable Memory Base and Limit Address registers define an Address range that the PCI 6520 uses to determine when to forward Memory transactions. The PCI 6520 forwards a Memory transaction from the primary-to-secondary interface, if the Transaction address falls within the Prefetchable Memory Address range. The PCI 6520 ignores Memory transactions initiated on the secondary interface that fall into this address range. The PCI 6520 does not respond to transactions that fall outside this address range on the primary interface and forwards those transactions upstream from the secondary interface (provided that the transactions do not fall into the Memory-Mapped I/O Address range, or are not forwarded by the VGA mechanism).

The PCI 6520 Prefetchable Memory range supports 64-bit addressing and provides additional registers to define the upper 32 bits of the Prefetchable Memory Base and Limit addresses. For address comparison, a Single Address Cycle (SAC; 32-bit address) Prefetchable Memory transaction is treated as a 64-bit Address transaction, where the upper 32 bits of the address are equal to 0h. This upper 32-bit value of 0h is compared to the Prefetchable Memory Base and Limit Address Upper 32 Bits registers. The Prefetchable Memory Base Address Upper 32 Bits register must be 0h to pass SAC transactions downstream.

The Prefetchable Memory Address range is defined by a 16-bit Prefetchable Memory Base Address register and a 16-bit Prefetchable Memory Limit Address register (PCIPMBAR; PCI:24h and PCIPMLMT; PCI:26h, respectively). The upper 12 bits of each of these registers correspond to bits [31:20] of the Memory address. The lower four Read-Only bits are hardcoded to 1h, indicating 64-bit address support. The lower 20 bits of the Prefetchable Memory Base address are assumed to be 0h, which results in a natural alignment to a 1-MB boundary. The lower

20 bits of the Prefetchable Memory Limit address are assumed to be F_FFFFh, which results in an alignment to the top of a 1-MB block. The maximum Memory Address range is 4 GB for 32-bit addressing, and 2^{64} bytes for 64-bit addressing.

Note: *Write the PCIPMBAR and PCIPMLMT registers with their appropriate values before setting the Command register Memory Space Enable or Master Enable bit.*

To disable 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 register value must be greater than the entire Limit register value (*that is*, 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 to a value greater than that of the lower Limit register; otherwise, the Upper 32-bit Base register must be greater than the Upper 32-bit Limit register.

10.4 ISA MODE

The PCI 6520 supports ISA mode by providing the Bridge Control register ISA Enable bit in Configuration space (BCNTRL[2]=1; PCI:3Eh). ISA mode modifies the PCI 6520 response inside the I/O Address range to support I/O space mapping in the presence of an ISA Bus in the system. This bit only affects the PCI 6520 response when the following conditions are met:

- Transaction falls inside the Address range defined by the I/O Base and Limit Address registers, and
- Address also falls inside the first 64 KB of I/O space (Address bits [31:16]=0h)

When the ISA Enable bit is set, the PCI 6520 does ***not*** forward downstream I/O transactions that address the upper 768 bytes of each aligned 1-KB block. Only those transactions addressing the lower 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 I/O Base and Limit register Address range.

Additionally, if the ISA Enable bit is set, the PCI 6520 forwards upstream those I/O transactions that address the upper 768 bytes of each aligned 1-KB block within the first 64 KB of I/O space. The Command Configuration register Master Enable bit must also be set (PCICR[2]=1; PCI:04h) to enable upstream forwarding. All other I/O transactions initiated on the secondary bus are forwarded upstream if the transactions fall outside the I/O Address range.

When the ISA Enable bit is set, devices downstream of the PCI 6520 can have I/O space mapped into the first 256 bytes of each 1-KB segment below the 64-KB boundary, or anywhere in I/O space above the 64-KB boundary.

10.5 VGA SUPPORT

The PCI 6520 provides two modes for VGA support:

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

10.5.1 VGA Mode

When a VGA-compatible device exists downstream from the PCI 6520, enable VGA mode by setting the Bridge Control register VGA Enable bit (BCNTRL[3]=1; PCI:3Eh). When operating in VGA mode, the PCI 6520 forwards downstream those transactions that address the VGA Frame Buffer Memory and VGA I/O registers, regardless of the I/O Base and Limit Address register values. The PCI 6520 ignores transactions initiated on the secondary interface addressing these locations.

The VGA Frame buffer resides in the Memory Address range—000A_0000h to 000B_FFFFh.

Read transactions to Frame Buffer memory are treated as non-prefetchable. The PCI 6520 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 I/O addresses 3B0h to 3BBh and 3C0h to 3DFh.

These I/O addresses are aliased every 1 KB throughout the first 64 KB of I/O space [*that is*, Address bits [15:10] are not decoded and can be any value, while Address bits [31:16] must be all zeros (0)].

VGA BIOS addresses starting at C_0000h are ***not*** decoded in VGA mode.

10.5.2 VGA Snoop Mode

The PCI 6520 provides VGA Snoop mode, allowing for VGA Palette Write transactions to be forwarded downstream. This mode is used when a graphics device downstream from the PCI 6520 must snoop or respond to VGA Palette Write transactions. To enable the mode, set the Command register VGA Palette Snoop Enable bit (PCICR[5]=1; PCI:04h). The PCI 6520 claims VGA Palette Write transactions by asserting DEVSEL# in VGA Snoop mode.

When the VGA Palette Snoop Enable bit is set, the PCI 6520 forwards downstream transactions with I/O addresses 3C6h, 3C8h, and 3C9h.

These addresses are also forwarded as part of the previously described VGA Compatibility mode. Again, Address bits [15:10] are ***not*** decoded, while Address bits [31:16] must be equal to 0h (*that is*, these addresses are aliased every 1 KB throughout the first 64 KB of I/O space).

Note: *If both the VGA Enable and VGA Palette Snoop Enable bits are set (BCNTRL[3]=1 and PCICR[5]=1, respectively), the PCI 6520 behaves as if only the VGA Enable bit is set.*

10.6 PRIVATE DEVICE SUPPORT

The PCI 6520 can support PCI devices that are not visible to primary port hosts or masters. These devices are referred to as *private devices* and occupy Private Memory space.

By pulling up the PRV_DEV input pin to 1, the PCI 6520 enables use of Private Memory space at power-up. The Private Memory range must be set up by driver software. The PCI 6520 does not respond to accesses to this Private Memory range on the primary or secondary port.

When PRV_DEV=1, in conjunction with the Private Memory range, secondary IDSELs using S_AD[23:16] are also re-routed to S_AD24. If S_AD24 is not connected to a PCI device, Type 1 accesses by the primary host to secondary PCI devices using S_AD[23:16] as IDSEL are Master Aborted on the secondary port.

The Private Device and Memory Enable bits (CCNTRL[3:2]; PCI:40h, respectively) are initialized by the PRV_DEV input pin state. However, software can change the values of these bits after reset.

11 TRANSACTION ORDERING

This section describes transaction ordering in Conventional PCI and PCI-X modes.

11.1 CONVENTIONAL PCI TRANSACTION ORDERING

This subsection describes the ordering rules that control Conventional PCI transaction forwarding across the PCI 6520. To maintain data coherency and consistency, the PCI 6520 complies with the ordering rules set forth in *PCI r2.3.* For a detailed discussion of transaction ordering, refer to *PCI r2.3,* Appendix E.

11.1.1 Transactions Governed by Ordering Rules

In Conventional PCI mode, ordering relationships are established for the following transaction classes that cross the PCI 6520:

- **Posted Write Transactions (Comprised of Memory Write, and Memory Write and Invalidate, Transactions)**—Completed at the source before completing at the destination (*that is*, data is written into intermediate Data buffers before reaching the target).
- **Delayed Write Request Transactions (Comprised of I/O Write and Configuration Write Transactions)**—Terminated by Target Retry on the initiator bus and queued in the Delayed Transaction queue. A Delayed Write transaction must complete on the target bus before completing on the initiator bus.
- **Delayed Write Completion Transactions (Comprised of I/O Write and Configuration Write Transactions)**—Completed on the target bus, with the target response queued in the buffers. A Delayed Write Completion transaction proceeds in the direction opposite to that of the original Delayed Write request (*that is*, the transaction proceeds from target-to-initiator bus).
- **Delayed Read Request Transactions (Comprised of all Memory Read, I/O Read, and Configuration Read Transactions)**—Terminated by Target Retry on the initiator bus and queued in the Delayed Transaction queue.
- **Delayed Read Completion Transactions (Comprised of all Memory Read, I/O Read, and Configuration Read Transactions)**—Completed on the target bus, and the Read data was queued in the Read Data buffers. A Delayed Read Completion transaction proceeds in the direction opposite that of the original Delayed Read request (*that is*, the transaction proceeds from target-to-initiator bus).

The PCI 6520 does ***not*** combine, merge, nor collapse Write transactions:

- **Combine separate Write transactions into a single Write transaction**—This optimization is best implemented in the originating master.
- **Merge bytes on separate Masked Write transactions to the same Dword address**—This optimization is also best implemented in the originating master.
- **Collapse sequential Write transactions to the same address into a single Write transaction**—*PCI r2.3* does not allow combining of transactions.

11.1.2 General Ordering Guidelines

Conventional PCI-independent transactions on the primary and secondary buses have a relationship only when those transactions cross the PCI 6520.

The following general ordering guidelines govern transactions crossing the PCI 6520:

- 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 a Target Retry can be accepted and completed in any order with respect to other transactions terminated with a 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 completes. If more than one Delayed transaction is initiated, the initiator should repeat all the Delayed transaction requests, using a fairness algorithm. Repeating a Delayed transaction ***cannot*** be contingent upon completion of another Delayed transaction; otherwise, deadlock may occur.

- Write transactions flowing in one direction have no ordering requirements with respect to Write transactions flowing in the other direction. The PCI 6520 can simultaneously accept Posted Write transactions on both interfaces, as well as simultaneously initiate Posted Write transactions on both interfaces.
- Acceptance of a Posted Memory Write transaction as a target can never be contingent on the completion of an Unlocked, Unposted transaction as a master. This is true of the PCI 6520 and must also be true of other bus agents; otherwise, deadlock may occur.
- PCI 6520 accepts Posted Write transactions, regardless of the state of completion of Delayed transactions being forwarded across the PCI 6520.

11.1.3 Ordering Rules

The following ordering rules describe the transaction relationships. Each ordering rule is followed by an explanation, and the ordering rules are referred to by number in Table 11-1. These ordering rules apply to Posted Write transactions, Delayed Write and Read requests, and Delayed Write and Read Completion transactions crossing the PCI 6520 in the same direction. Note that Delayed Completion transactions cross the PCI 6520 in the direction opposite that of the corresponding Delayed requests.

1. **Posted Write**—Posted Write transactions must complete on the target bus in the order in which the transactions were received on the initiator bus.

   The subsequent Posted Write transaction could 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, devices checking that flag could subsequently be using stale data.
2. **Delayed Write Request**—Delayed Write requests ***cannot*** pass previously queued Posted Write data. As in the case of Posted Memory Write transactions, the Delayed Write transaction might be setting a flag regarding data in the Posted Write transaction. If the Delayed Write request were to complete before the earlier Posted Write transaction, devices checking the flag could subsequently be using stale data.
3. **Delayed Read Request**—Delayed Read requests traveling in the same direction as previously queued Posted Write transactions 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 might be in the same location as the Write data; therefore, if the Read transaction were to pass the Write transaction, the read would return stale data.
4. **Delayed Write Completion**—Posted Write transactions must be provided opportunities to pass Delayed Read and Write requests and completions. Otherwise, deadlock 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 and Delayed Transaction queues.

   The PCI 6520 can return Delayed Read transactions in a different order than requested if the DRT Out-of-Order Enable bit is set to 1 (MSCOPT[2]=1; PCI:46h). Requested cycles can execute out of order across the bridge, if all other ordering rules are satisfied. Therefore, if the PCI 6520 starts a Delayed transaction that is Retried by the target, the PCI 6520 can execute another transaction in the Delayed Transaction Request queue. Also, if there are Delayed Write and Read requests in the queue, and the Read Data FIFOs are full, the PCI 6520 may execute the Delayed Write request before the Delayed Read request.
5. **Delayed Read Completion**—Delayed Read completions 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 the bridge as the target of the Write transaction. The Posted Write transaction must complete on the target before Read data is returned to the initiator.

   The Read transaction could be to a Status register of the initiator of the Posted Write data and therefore should not complete until the Write transaction is complete.

The PCI 6520 can generate cycles across the bridge in the same order requested if the Miscellaneous Options register DRT Out-of-Order Enable bit is set (MSCOPT[2]=1; PCI:46h). By default, requested cycles can execute out of order across the bridge if all other ordering rules are satisfied. Therefore, if the PCI 6520 begins a Delayed transaction that is Retried by the target, the PCI 6520 can execute another transaction in the Delayed Transaction Request queue. Additionally, if there is both Delayed Write and Delayed Read requests in the queue, and the Read Data FIFO is full, the PCI 6520 may execute the Delayed Write request before the Delayed Read request.

On cycle completion, the PCI 6520 may complete cycles in a different order than that requested by the initiator.

11.1.4 Data Synchronization

Data synchronization refers to the relationship between interrupt signaling and data delivery. *PCI r2.3* provides the following alternative methods for synchronizing data and interrupts:

- Device signaling the interrupt performs a read of the data just written (software)
- Device driver performs a Read operation to any register in the interrupting device before accessing data written by the device (software)
- System hardware guarantees that Write buffers are flushed before interrupts are forwarded

The PCI 6520 does not have a hardware mechanism to guarantee data synchronization for Posted Write transactions. Therefore, all Posted Write transactions must be followed by a Read operation, from the PCI 6520 to the location recently written (or some other location along the same path), or from the device driver to a PCI 6520 register.

Table 11-1. Conventional PCI Transaction Ordering Summary

Pass	Posted Write	Delayed Write Request	Delayed Read Request	Delayed Write Completion	Delayed Read Completion
Posted Write	N[1]	Y[4]	Y[4]	Y[4]	Y[4]
Delayed Write Request	N[5]	Y	Y	Y	Y
Delayed Read Request	N[3]	Y	Y	Y	Y
Delayed Write Completion	Y	Y	Y	Y	Y
Delayed Read Completion	N[2]	Y	Y	Y	Y

***Legend**:*

***Superscript Number** = Refers to the five applicable ordering rules listed in Section 11.1.3. Many entries are not governed by these ordering rules; therefore, the implementation can choose whether the transactions pass each other.*

***Y** = Transactions may be completed out of order or "pass" each other.*

***N** = Row transaction must **not** pass the column transaction.*

11.2 PCI-X TRANSACTION ORDERING

This subsection describes the ordering rules that control PCI-X transaction forwarding across the PCI 6520. For a detailed discussion of transaction ordering, refer to *PCI-X r1.0b,* Use of Relaxed Ordering Appendix, Chapter 11.

PCI-X introduces two features that affect transaction ordering and passing rules that are not present in Conventional PCI—Relaxed Ordering Attribute bit and Split transactions.

11.2.1 Relaxed Ordering Attribute Bit

The PCI-X Relaxed Ordering Attribute bit (AD29) may be used to allow a Memory Write transaction to pass other Memory Writes and to allow Split Read Completions to pass Memory Writes.

- If the Relaxed Ordering Attribute bit is set for a Read transaction, the completion for that transaction is allowed to pass previously Posted Memory Write transactions traveling in the direction of the completion.
- If the Relaxed Ordering Attribute bit is set for a Memory Write transaction, that transaction is allowed to pass previous Posted Memory Write transactions moving in the same direction on the host bridge.

Note: *The PCI 6520 ignores the Relaxed Ordering Attribute bit for Memory Write transactions and maintains the order of Memory Write transactions that cross the PCI 6520.*

11.2.2 Split Transactions

In PCI-X, Split Transaction ordering and Deadlock-Avoidance rules are almost identical to the Delayed Transaction rules in Conventional PCI (transaction order is established as the transactions complete). Table 11-2 lists the ordering requirements for all Split and Memory Write transactions (the columns represent the first of two transactions, and the rows represent the second). Table 11-3 provides a case-by-case discussion of Split transactions.

- Split Requests can be reordered with respect to other Split Requests. If an initiator requires two Split transactions to complete in order, the initiator must not issue the second request until the first Split transaction completes.
- Split Read Completions that have the same Sequence ID must remain in address order. The completer must supply the Split Read Completions on the bus, in address order, to guarantee that the requester always receives the data in its natural order. There are no ordering restrictions for Split Read Completions with different Sequence IDs.

Table 11-2. PCI-X Transaction Ordering and Deadlock-Avoidance Rules

Row Pass Column?	Memory Write (Column 2)	Split Read Request (Column 3)	Split Write Request (Column 4)	Split Read Completion (Column 5)	Split Write Completion (Column 6)
Memory Write (Row A)	a) No b) Y/N	Yes	Yes	Yes	Yes
Split Read Request (Row B)	No	Y/N	Y/N	Y/N	Y/N
Split Write Request (Row C)	No	Y/N	Y/N	Y/N	Y/N
Split Read Completion (Row D)	a) No b) Y/N	Yes	Yes	a) Y/N b) No	Y/N
Split Write Completion (Row E)	Y/N	Yes	Yes	Y/N	Y/N

***Legend**:*

***Yes** = Second transaction must be allowed to pass the first transaction to avoid deadlock.*

***Y/N** = PCI 6520 may optionally allow the second transaction to pass the first.*

***No** = Second transaction must not be allowed to pass the first transaction (to preserve strong write ordering).*

***Note:** Refer to Table 11-3 for a case-by-case discussion of the Split transactions.*

Table 11-3. PCI-X Split Transactions—Case-by-Case Discussion

Split Transaction	Comments
A2a	Unless the Relaxed Ordering Attribute bit is set, a Memory Write transaction must not pass any other Memory Writes.
A2b	If the Relaxed Ordering attribute bit is set, that Memory Write transaction is allowed to pass all other Memory Writes in the host bridge only.
A3, A4	Memory Write transactions must be allowed to pass Split Requests to avoid deadlock.
A5, A6	Memory Write transactions must be allowed to pass Split Completions to avoid deadlock.
B2, C2	Split Requests ***cannot*** pass Memory Write transactions.
B3, B4	Split Requests are allowed to be blocked by, or pass, Split Completions.
C3, C4	Split transactions have the same requirements as Conventional PCI Delayed transactions.
B5, B6	Split Requests are allowed to be blocked by, or pass, Split Completions.
C5, C6	Split Requests pass Split Completions. (Deadlock does not occur if Split Completions block Split Requests.)
D2a	Unless the Relaxed Ordering Attribute bit is set, Split Read Completions ***cannot*** pass Memory Writes.
D2b	If the Relaxed Ordering Attribute bit is set, Split Read Completions are allowed to pass previous Posted Memory Write transactions.
D3, D4	Split Read Completions must be allowed to pass all Split Requests to avoid deadlock.
D5a	Unless two Split Read Completions are part of the same Sequence, they are allowed to be blocked by, or pass, each other.
D5b	Split Read Completions with the same Sequence ID must remain in address order.
D6	Split Read Completions are allowed to be blocked by, or pass, Split Write Completions.
E2	Split Write Completions are allowed to be blocked by, or pass, Memory Write transactions moving in the opposite direction (they have no ordering relationship).
E3, E4	Split Write Completions must be allowed to pass all Split Requests to avoid deadlock.
E5, E6	Split Write Completions are allowed to be blocked by, or pass, Split Read and Write Completions.

12 ERROR HANDLING

This section provides detailed information regarding PCI 6520 error management. It also describes error status reporting and error operation disabling.

12.1 OVERVIEW

The PCI 6520 checks, forwards, and generates parity on the primary and secondary interfaces. To maintain transparency, the PCI 6520 can either forward the existing parity condition from one bus to the other, along with address and data, or regenerate the data parity on the other bus.

To support error reporting on the PCI Bus, the PCI 6520 implements the following:

- P_PERR#, P_SERR#, S_PERR#, and S_SERR# signals
- Primary and secondary Status registers (PCISR; PCI:06h and PCISSR; PCI:1Eh, respectively)
- Device-specific P_SERR# Event Disable and Status registers (PSERRED; PCI:64h and PSERRSR; PCI:6Ah, respectively)

12.2 ADDRESS PARITY ERRORS

The PCI 6520 checks address parity for all Bus transactions, and Address and Bus commands.

When the PCI 6520 detects an Address Parity error on the primary interface, the following occurs:

1. **Conventional PCI mode**—If the Command register Parity Error Response Enable bit is set (PCICR[6]=1; PCI:04h), the PCI 6520 does not claim the transaction with P_DEVSEL#. If the Parity Error Response Enable bit is ***not*** set, the PCI 6520 proceeds as usual and accepts the transaction if the transaction is directed to, or across, the PCI 6520.

   PCI-X mode—With a PCI-X Split Completion command, the PCI 6520 receives the cycle as if there is no Parity error. With a PCI-X command other than Split Completion, the PCI 6520 claims the bus as usual and immediately terminates the cycle with a Target Abort. In either case, the PCI 6520 does ***not*** accept the cycle in its FIFO operation.
2. PCI 6520 sets the Status register Parity Error Detected bit (PCISR[15]=1; PCI:06h).
3. PCI 6520 asserts P_SERR# and sets the Status register Signaled System Error bit (PCISR[14]=1), if the Command register P_SERR# Enable and Parity Error Response Enable bits are set (PCICR[8, 6]=11b; PCI:04h).

When the PCI 6520 detects an Address Parity error on the secondary interface, the following occurs:

1. **Conventional PCI mode**—If the Bridge Control register Parity Error Response Enable bit is set (BCNTRL[0]=1; PCI:3Eh), the PCI 6520 does not claim the transaction with S_DEVSEL#. If the Parity Error Response Enable bit is ***not*** set, the PCI 6520 proceeds as usual and accepts the transaction if the transaction is directed to, or across, the PCI 6520.

   PCI-X mode—With a PCI-X Split Completion command, the PCI 6520 receives the cycle as if there is no Parity error. With a PCI-X command other than Split Completion, the PCI 6520 claims the bus as usual and immediately terminates the cycle with a Target Abort. In either case, the PCI 6520 does ***not*** accept the cycle in its FIFO operation.
2. PCI 6520 sets the Secondary Status register Parity Error Detected bit (PCISSR[15]=1; PCI:1Eh), regardless of the Parity Error Response Enable bit state (PCICR[6]=x).
3. PCI 6520 asserts S_SERR# and sets the Status register Signaled System Error bit (PCISSR[14]=1).

12.3 ATTRIBUTE PARITY ERRORS—PCI-X MODE

PCI-X Attribute Parity errors are managed in the same way as Address Parity errors. (Refer to Section 12.2.)

12.4 DATA PARITY ERRORS

When forwarding transactions, the PCI 6520 attempts to pass the data parity condition from one interface to the other unchanged, whenever possible, to allow the master and target devices to manage the error condition.

The following subsections describe, for each transaction, the sequence that occurs when a Parity error is detected and the way in which the parity condition is forwarded across the bridge.

12.4.1 Configuration Write Transactions to Configuration Space

When the PCI 6520 detects a Data Parity error during a Type 0 Configuration Write transaction to Configuration space, the following occurs:

1. If the Command register Parity Error Response Enable bit is set (PCICR[6]=1; PCI:04h), the PCI 6520 asserts P_PERR#. If the Parity Error Response Enable bit is ***not*** set, the PCI 6520 does ***not*** assert P_PERR#. In either case, the Configuration register is written.
2. PCI 6520 sets the Status register Parity Error Detected bit (PCISR[15]=1; PCI:06h), regardless of the Parity Error Response Enable bit state (PCICR[6]=x).

12.4.2 Read Transactions

When the PCI 6520 detects a Parity error during a Read transaction, the target drives data and data parity, and the initiator checks parity and conditionally asserts P_PERR# or S_PERR#.

For downstream transactions, when the PCI 6520 detects a Read Data Parity error on the secondary bus, the PCI 6520:

1. Asserts S_PERR# two cycles following the Data transfer, if the secondary interface Bridge Control register Parity Error Response Enable bit is set (BCNTRL[0]=1; PCI:3Eh).
2. Sets the secondary Status register Parity Error Detected bit (PCISSR[15]=1; PCI:1Eh), regardless of the Parity Error Response Enable bit state (PCICR[6]=x).
3. Sets the secondary Status register Data Parity Error Detected bit (PCISSR[8]=1), if BCNTRL[0]=1.
4. Returns the bad parity with the data to the initiator on the primary bus. If the data with the bad parity is prefetched and not read by the initiator on the primary bus, the data is discarded and data with bad parity is not returned to the initiator.
5. Completes the transaction as usual.

For upstream transactions, when the PCI 6520 detects a Read Data Parity error on the primary bus, the PCI 6520:

1. Asserts P_PERR# two cycles following the Data transfer, if the primary interface Command register Parity Error Response Enable bit is set (PCICR[6]=1).
2. Sets the primary Status register Parity Error Detected bit (PCISR[15]=1).
3. Sets the primary Status register Data Parity Error Detected bit (PCISR[8]=1), if PCICR[6]=1.
4. Returns the bad parity with the data to the initiator on the secondary bus. If the data with the bad parity is prefetched and not read by the initiator on the secondary bus, the data is discarded and data with bad parity is not returned to the initiator.
5. Completes the transaction as usual.

The PCI 6520 returns to the initiator the data and parity received from the target. When the initiator detects a Parity error on this Read data and is enabled to report the error, the initiator asserts its PERR# signal (which is then connected to the PCI 6520 P_PERR# or S_PERR# signal, depending on the initiator bus) two cycles after the Data transfer. It is assumed that the initiator is responsible for handling Parity error conditions; therefore, when the PCI 6520 detects the initiator's PERR# assertion while returning Read data to the initiator, the PCI 6520 takes no further action and completes the transaction as usual.

12.4.3 Posted Write Transactions

During downstream Posted Write transactions, when the PCI 6520 is responding as a target and detects a Data Parity error on the initiator (primary) bus, it:

1. Asserts P_PERR# two cycles after the Data transfer, if the primary interface Command register Parity Error Response Enable bit is set (PCICR[6]=1).
2. Sets the primary interface Status register Parity Error Detected bit (PCISR[15]=1).
3. Captures and forwards the bad parity condition to the secondary bus.
4. Completes the transaction as usual.

Similarly, during upstream Posted Write transactions, when the PCI 6520 is responding as a target and detects a Data Parity error on the initiator (secondary) bus, it:

1. Asserts S_PERR# two cycles after the Data transfer, if the secondary interface Bridge Control register Parity Error Response Enable bit is set (BCNTRL[0]=1).
2. Sets the secondary interface Status register Parity Error Detected bit (PCISSR[15]=1).
3. Captures and forwards the bad parity condition to the primary bus.
4. Completes the transaction as usual.

During downstream Write transactions, when a Data Parity error is reported on the target (secondary) bus by the target's assertion of S_PERR#, the PCI 6520:

1. Sets the secondary Status register Data Parity Error Detected bit (PCISSR[8]=1), if the secondary interface Bridge Control register Parity Error Response Enable bit is set (BCNTRL[0]=1).
2. Asserts P_SERR# and sets the Status register Signaled System Error bit (PCISR[14]=1), if the following conditions are met:
   - Primary interface Command register P_SERR# Enable and Parity Error Response Enable bits are set (PCICR[8, 6]=11b, respectively), and
   - Device-specific P_SERR# Disable bit for Posted Write Parity errors is ***not*** set (PSERRED[1]=0; PCI:64h), and
   - Secondary interface Bridge Control register Parity Error Response Enable bit is set (BCNTRL[0]=1), and
   - PCI 6520 did not detect the Parity error on the initiator (primary) bus (*that is*, the Parity error was not forwarded from the 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#, the PCI 6520:

1. Sets the Status register Data Parity Error Detected bit (PCISR[8]=1), if the primary interface Command register Parity Error Response Enable bit is set (PCICR[6]=1).
2. Asserts P_SERR# and sets the Status register Signaled System Error bit (PCISR[14]=1), if the following conditions are met:
   - Primary interface Command register P_SERR# Enable and Parity Error Response Enable bits are set (PCICR[8, 6]=11b, respectively), and
   - Secondary interface Bridge Control register Parity Error Response Enable bit is set (BCNTRL[0]=1), and
   - PCI 6520 did not detect the Parity error on the initiator (secondary) bus (*that is*, the Parity error was not forwarded from the secondary bus)

P_SERR# assertion signals the Parity error condition when the initiator is not sent information about an error having occurred. Because the data is delivered with no errors, there is no other way to signal this information to the initiator.

If a Parity error is forwarded from the initiator bus to the target bus, P_SERR# is ***not*** asserted.

12.4.4 Delayed Write Transactions

When the PCI 6520 detects a Data Parity error during a Delayed Write transaction, it conditionally asserts PERR#. The PCI 6520 either passes or regenerates data parity to the target bus. A Parity error can occur:

- During the original Delayed Write Request transaction
- When the initiator repeats the Delayed Write Request transaction
- When the PCI 6520 completes the Delayed Write transaction to the target

12.4.4.1 Conventional PCI Mode

In Conventional PCI mode, when a Delayed Write transaction is queued, the Address, Command, Address and Data Parity, Data, and Byte Enable bits are captured and a Target Retry is returned to the initiator. When the PCI 6520 detects a Parity error on the Write data for the initial Delayed Write Request transaction, the following occurs:

1. If the Parity Error Response Enable bit corresponding to the initiator bus is set (primary—PCICR[6]=1, secondary—BCNTRL[0]=1), the PCI 6520 asserts P_PERR# or S_PERR# two

clocks after the data. The PCI 6520 always accepts the cycle, and can optionally pass the incorrect parity to the other bus, or regenerate the Parity bit on the other bus (MSCOPT[3]=1; PCI:46h).

2. PCI 6520 sets the Status register Parity Error Detected bit corresponding to the initiator bus (primary—PCISR[15]=1, secondary—PCISSR[15]=1), regardless of the Parity Error Response Enable bit state (PCICR[6]=x).

Following the initiating transaction (the first PCI 6520 Retry), the subsequent Data Parity error of a similar transaction on the initiating bus is detected as usual; however, the Data Parity error no longer affects FIFO operation. The cycles are considered similar if they have the same Address, Command, Byte Enables and Write data. The Parity bit is not part of this "similar" detection operation. Therefore, if a Data Parity error occurs only in the Parity bit (same data as before), the cycle operates as usual. Conversely, if a Data Parity error occurs in the data segment (different data from the initiating Write data), the PCI 6520 treats the error as a new Delayed Write transaction.

12.4.4.2 PCI-X Mode

In PCI-X mode, a Delayed Write transaction is queued with its Address, Command, Parity, Data, and Byte Enable bits, then returns a Split Response to the initiator. When the PCI 6520 detects a Parity error on the Write data, the following occurs:

1. If the Parity Error Response Enable bit corresponding to the initiator bus is set (primary—PCICR[6]=1, secondary—BCNTRL[0]=1), the PCI 6520 asserts P_PERR# or S_PERR#. The PCI 6520 always accepts the cycle, and can optionally pass the incorrect parity to the other bus, or regenerate the Parity bit on the other bus (MSCOPT[3]=1; PCI:46h).
2. PCI 6520 sets the Status register Parity Error Detected bit corresponding to the initiator bus (primary—PCISR[15]=1, secondary—PCISSR[15]=1), regardless of the Parity Error Response Enable bit state (PCICR[6]=x).

12.4.5 Split Completion—PCI-X Mode

When detecting a Data Parity error on the originating bus for a Split Completion other than a Split Completion Message, the PCI 6520 asserts P_PERR# or S_PERR# and sets the appropriate Error Status bit for that interface. The PCI 6520 "drives bad parity" when it forwards the Split Completion.

When the PCI 6520 detects a Data Parity error on the originating bus for a Split Completion Message, the following occurs:

- **PCI-X-to-PCI mode**—Parity error is not considered in the FIFO operation. If the Split Completion Error bit is set to 1 (AD30=1), the PCI 6520 signals a Target Abort; otherwise, it is considered to be a normal completion.
- **PCI-X-to-PCI-X mode**—PCI 6520 forwards the exact message with the incorrect Parity bit to the other bus.

12.5 DATA PARITY ERROR REPORTING SUMMARY

In the previous subsections, the PCI 6520 responses to Data Parity errors are presented according to transaction type in progress. This subsection organizes the PCI 6520 responses to Data Parity errors according to the Status bits set by the PCI 6520 and the signals asserted.

Table 12-1 delineates the primary interface Status register Parity Error Detected bit status. This bit is set when the PCI 6520 detects a Parity error on the primary interface.

Table 12-2 delineates the secondary interface Status register Parity Error Detected bit status. This bit is set when the PCI 6520 detects a Parity error on the secondary interface.

Table 12-3 delineates the primary interface Status register Data Parity Error Detected bit status. This bit is set under the following conditions:

- PCI 6520 must be a master on the primary bus, and
- Primary interface Command register Parity Error Response Enable bit must be set (PCICR[6]=1)

Table 12-4 delineates the secondary interface Status register Data Parity Error Detected bit status. This bit is set under the following conditions:

- PCI 6520 must be a master on the secondary bus, and
- Secondary interface Bridge Control register Parity Error Response Enable bit must be set (BCNTRL[0]=1)

Table 12-5 delineates P_PERR# assertion. This signal is set under the following conditions:

- PCI 6520 is either the target of a Write transaction or the initiator of a Read transaction on the primary bus, and
- Primary interface Command register Parity Error Response Enable bit must be set (PCICR[6]=1), and
- PCI 6520 detects a Data Parity error on the primary bus, or detects S_PERR# asserted during the Completion phase of a downstream Delayed Write transaction on the target (secondary) bus

Table 12-6 delineates S_PERR# assertion. This signal is set under the following conditions:

- PCI 6520 is either the target of a Write transaction or the initiator of a Read transaction on the secondary bus, and
- Secondary interface Bridge Control register Parity Error Response Enable bit must be set (BCNTRL[0]=1), and
- PCI 6520 detects a Data Parity error on the secondary bus, or detects P_PERR# asserted during the Completion phase of an upstream Delayed Write transaction on the target (primary) bus

Table 12-7 delineates P_SERR# or S_SERR# assertion. This signal is set under the following conditions:

- Command register P_SERR# Enable and Parity Error Response Enable bits must be set (PCICR[8, 6]=11b, respectively),
- Bridge Control register Parity Error Response Enable bit must be set (BCNTRL[0]=1)

Table 12-1. Primary Interface Parity Error Detected Bit Status

Primary Parity Error Detected Bit (PCISR[15])	Transaction Type	Direction	Bus on which Error Detected	Primary Parity Error Response Enable Bit (PCICR[6])	Secondary Parity Error Response Enable Bit (BCNTRL[0])
0	Read	Downstream	Primary	x	x
0			Secondary	x	x
1		Upstream	Primary	x	x
0			Secondary	x	x
1	Posted Write	Downstream	Primary	x	x
0			Secondary	x	x
0		Upstream	Primary	x	x
0			Secondary	x	x
1	Delayed Write	Downstream	Primary	x	x
0			Secondary	x	x
0		Upstream	Primary	x	x
0			Secondary	x	x

Note: *x = Don't care.*

Table 12-2. Secondary Interface Parity Error Detected Bit Status

Secondary Parity Error Detected Bit (PCISSR[15])	Transaction Type	Direction	Bus on which Error Detected	Primary Parity Error Response Enable Bit (PCICR[6])	Secondary Parity Error Response Enable Bit (BCNTRL[0])
0	Read	Downstream	Primary	x	x
1			Secondary	x	x
0		Upstream	Primary	x	x
0			Secondary	x	x
0	Posted Write	Downstream	Primary	x	x
0			Secondary	x	x
0		Upstream	Primary	x	x
1			Secondary	x	x
0	Delayed Write	Downstream	Primary	x	x
0			Secondary	x	x
0		Upstream	Primary	x	x
1			Secondary	x	x

Note: *x = Don't care.*

Table 12-3. Primary Interface Data Parity Error Detected Bit Status

Primary Data Parity Error Detected Bit (PCISR[8])	Transaction Type	Direction	Bus on which Error Detected	Primary Parity Error Response Enable Bit (PCICR[6])	Secondary Parity Error Response Enable Bit (BCNTRL[0])
0	Read	Downstream	Primary	x	x
0			Secondary	x	x
1		Upstream	Primary	1	x
0			Secondary	x	x
0	Posted Write	Downstream	Primary	x	x
0			Secondary	x	x
1		Upstream	Primary	1	x
0			Secondary	x	x
0	Delayed Write	Downstream	Primary	x	x
0			Secondary	x	x
1		Upstream	Primary	1	x
0			Secondary	x	x

Note: *x = Don't care.*

Table 12-4. Secondary Interface Data Parity Error Detected Bit Status

Secondary Data Parity Error Detected Bit (PCISSR[8])	Transaction Type	Direction	Bus on which Error Detected	Primary Parity Error Response Enable Bit (PCICR[6])	Secondary Parity Error Response Enable Bit (BCNTRL[0])
0	Read	Downstream	Primary	x	x
1			Secondary	x	1
0		Upstream	Primary	x	x
0			Secondary	x	x
0	Posted Write	Downstream	Primary	x	x
1			Secondary	x	1
0		Upstream	Primary	x	x
0			Secondary	x	x
0	Delayed Write	Downstream	Primary	x	x
1			Secondary	x	1
0		Upstream	Primary	x	x
0			Secondary	x	x

Note: *x = Don't care.*

Table 12-5. P_PERR# Assertion

P_PERR#	Transaction Type	Direction	Bus on which Error Detected	Primary Parity Error Response Enable Bit (PCICR[6])	Secondary Parity Error Response Enable Bit (BCNTRL[0])
1 (De-asserted)	Read	Downstream	Primary	x	x
1			Secondary	x	x
0 (Asserted)		Upstream	Primary	1	x
1			Secondary	x	x
0	Posted Write	Downstream	Primary	1	x
1			Secondary	x	x
1		Upstream	Primary	x	x
1			Secondary	x	x
0	Delayed Write	Downstream	Primary	1	x
0*			Secondary	1	1
1		Upstream	Primary	x	x
1			Secondary	x	x

Notes: *x = Don't care.*

** Parity error detected on the target (secondary) bus, but not on the initiator (primary) bus.*

Table 12-6. S_PERR# Assertion

S_PERR#	Transaction Type	Direction	Bus on which Error Detected	Primary Parity Error Response Enable Bit (PCICR[6])	Secondary Parity Error Response Enable Bit (BCNTRL[0])
1 (De-asserted)	Read	Downstream	Primary	x	x
0 (Asserted)			Secondary	x	1
1		Upstream	Primary	x	x
1			Secondary	x	x
1	Posted Write	Downstream	Primary	x	x
1			Secondary	x	x
1		Upstream	Primary	x	x
0			Secondary	x	1
1	Delayed Write	Downstream	Primary	x	x
1			Secondary	x	x
0*		Upstream	Primary	1	1
0			Secondary	x	1

Note: *x = Don't care.*

** Parity error detected on the target (secondary) bus, but not on the initiator (primary) bus.*

Table 12-7. P_SERR# or S_SERR# for Data Parity Error Assertion

P_SERR# or S_SERR#	Transaction Type	Direction	Bus on which Error Detected	Primary Parity Error Response Enable Bit (PCICR[6])	Secondary Parity Error Response Enable Bit (BCNTRL[0])
1 (De-asserted)	Read	Downstream	Primary	x	x
1			Secondary	x	x
1		Upstream	Primary	x	x
1			Secondary	x	x
1	Posted Write	Downstream	Primary	x	x
0* (Asserted)			Secondary	1	1
0**		Upstream	Primary	1	1
1			Secondary	x	x
1	Delayed Write	Downstream	Primary	x	x
1			Secondary	x	x
1		Upstream	Primary	x	x
1			Secondary	x	x

Notes: *x = Don't care.*

** Parity error detected on the target (secondary) bus, but not on the initiator (primary) bus.*

*** Parity error detected on the target (primary) bus, but not on the initiator (secondary) bus.*

12.6 SYSTEM ERROR (P_SERR#) REPORTING

The PCI 6520 uses the P_SERR# signal to conditionally report a number of System error conditions in addition to the special case Parity error conditions.

In this data book, when P_SERR# assertion is discussed, the following conditions are assumed:

- For the PCI 6520 to assert P_SERR#, the Command register P_SERR# Enable bit must be set (PCICR[8]=1)
- When the PCI 6520 asserts P_SERR#, the PCI 6520 must also set the Status register Signaled System Error bit (PCISR[14]=1)

In compliance with *P-to-P Bridge r1.2*, the PCI 6520 asserts P_SERR# when it detects S_SERR# input asserted and the Bridge Control register S_SERR# Enable bit is set (BCNTRL[1]=1). In addition, the PCI 6520 also sets the secondary Status register Signaled System Error bit (PCISSR(14]=1).

The PCI 6520 also conditionally asserts P_SERR# for the following conditions:

- Master Abort detected during Posted Write transaction (on the secondary bus)
- Target Abort detected during Posted Write transaction (on the secondary bus)
- Posted Write data discarded after 2^{24} delivery attempts (2^{24} Target Retries received)
- S_PERR# reported on the target bus during a Posted Write transaction (refer to Section 12.5)
- Delayed Write data discarded after 2^{24} delivery attempts (2^{24} Target Retries received)
- Delayed Read data ***cannot*** be transferred from the target after 2^{24} attempts (2^{24} Target Retries received)
- Master Timeout on Delayed transaction

The device-specific P_SERR# Status register reports the reason for P_SERR# assertion.

Most of these events have additional device-specific Disable bits in the P_SERR# Event Disable register that can mask P_SERR# assertion for specific events. The Master Timeout condition has S_SERR# and P_SERR# Enable bits for that event in the Bridge Control register (BCNTRL[12:11], respectively), and therefore does not have a device-specific Disable bit.

13 EXCLUSIVE ACCESS

This section describes P_LOCK# and S_LOCK# signal use to implement exclusive access to a target for transactions crossing the PCI 6520, including concurrent locks, and acquiring and ending exclusive access.

13.1 CONCURRENT LOCKS

The primary and secondary bus Lock mechanisms concurrently operate, *except* when a Locked transaction is crossing the PCI 6520. A primary master can lock a primary target without affecting the Lock status on the secondary bus, and vice versa. This means that a primary master can lock a primary target concurrent with a secondary master locking a secondary target.

13.2 ACQUIRING EXCLUSIVE ACCESS ACROSS PCI 6520

For a PCI Bus, before acquiring access to the P_LOCK# and/or S_LOCK# signal and starting a series of Locked transactions, the initiator must first verify whether the following conditions are met:

- PCI Bus is idle, and
- P_LOCK# and/or S_LOCK# is de-asserted

The initiator leaves P_LOCK# and/or S_LOCK# de-asserted during the Address phase and asserts P_LOCK# and/or S_LOCK# one Clock cycle later. Target lock is achieved after the target completes a Data transfer.

Locked transactions can cross the PCI 6520 in the downstream and upstream directions, from the primary-to-secondary bus and vice versa.

When the target resides on another PCI Bus, the master must acquire not only the lock on its own PCI Bus, but also the lock on every bus between its bus and the target bus. When the PCI 6520 detects an initial Locked transaction on the primary bus that is intended for a target on the secondary bus, the PCI 6520 samples the Address, Transaction Type, Byte Enable, and Parity bits, and the S_LOCK# signal. Because a Target Retry is signaled to the initiator, the initiator must relinquish the lock on the primary bus, and therefore the lock is not yet established.

The first Locked transaction must be a Read transaction. Subsequent Locked transactions can be Write or Read transactions. Posted Memory Write transactions that are part of the Locked-transaction sequence are nevertheless posted. Memory Read transactions that are part of the Locked-transaction sequence are ***not*** prefetched.

When the Locked Delayed Read request is queued, the PCI 6520 does ***not*** queue further transactions until the locked sequence is complete. The PCI 6520 signals a Target Retry to all transactions initiated subsequent to the Locked Read transaction that are intended for targets on the opposite side of the PCI 6520. The PCI 6520 allows transactions queued before the Locked transaction to complete before initiating the Locked transaction.

When the Locked Delayed Read request moves to the head of the Delayed Transaction queue, the PCI 6520 initiates the request as a Locked Read transaction by de-asserting S_LOCK# on the target bus during the first Address phase, then re-asserting S_LOCK# one cycle later. If S_LOCK# was previously asserted (used by another initiator), the PCI 6520 waits to request access to the secondary bus until S_LOCK# is sampled de-asserted when the target bus is idle. Note that the existing lock on the target bus did not cross the PCI 6520; otherwise, the pending queued Locked transaction would not have queued. When the PCI 6520 is able to complete a Data transfer with the Locked Read transaction, the lock is established on the secondary bus.

When the initiator repeats the Locked Read transaction on the primary bus with the same Address, Transaction Type, Byte Enable, and Parity bits, the PCI 6520 transfers the Read data back to the initiator, and the lock is also established on the primary bus.

For the PCI 6520 to recognize and respond to the initiator, the initiator's subsequent Read transaction attempts must use the Locked-transaction sequence (de-assert P_LOCK# during the Address phase, then re-assert P_LOCK# one cycle later). If the P_LOCK# sequence is not used in subsequent attempts, a Master Timeout condition may result. When a Master Timeout condition occurs, P_SERR# is conditionally

asserted, the Read data and queued Read transaction are discarded, and S_LOCK# is de-asserted on the target bus.

After the intended target is locked, subsequent Locked transactions initiated on the initiator bus that are forwarded by the PCI 6520 are driven as Locked transactions on the target bus.

When the PCI 6520 receives a Master or Target Abort in response to the Delayed Locked Read transaction, this status is passed back to the initiator, and no locks are established on the initiator or target bus. The PCI 6520 resumes Unlocked transaction forwarding in both directions.

13.3 ENDING EXCLUSIVE ACCESS

After the lock is acquired on the initiator and target buses, the PCI 6520 must maintain the lock on the target bus for subsequent Locked transactions until the initiator relinquishes the lock.

The only time a Target Retry causes the lock to be relinquished is on the first transaction of a Locked sequence. On subsequent transactions in the sequence, the Target Retry has no effect on the P_LOCK# and/or S_LOCK# signal status.

An established target lock is maintained until the initiator relinquishes the lock. The PCI 6520 does not recognize whether the current transaction is the last in a sequence of Locked transactions until the initiator de-asserts P_LOCK# and/or S_LOCK# at the end of the transaction.

When the last Locked transaction is a Delayed transaction, the PCI 6520 previously completed the transaction on the secondary bus. In this case, when the PCI 6520 detects that the initiator has relinquished the P_LOCK# and/or S_LOCK# signal by sampling the signal de-asserted while P_FRAME# or S_FRAME# is de-asserted, the PCI 6520 de-asserts P_LOCK# and/or S_LOCK# on the target bus when possible. Because of this behavior, P_LOCK# and/or S_LOCK# may not be de-asserted until several cycles after the last Locked transaction completes on the target bus. After de-asserting P_LOCK# and/or S_LOCK# to indicate the end of a sequence of Locked transactions, the PCI 6520 resumes Unlocked transaction forwarding.

When the last Locked transaction is a Posted Write, the PCI 6520 de-asserts P_LOCK# and/or S_LOCK# on the target bus at the end of the transaction because the lock was relinquished at the end of the Write transaction on the initiator bus.

When the PCI 6520 receives a Master or Target Abort in response to a Locked Delayed transaction, the PCI 6520 returns a Master or Target Abort when the initiator repeats the Locked transaction. The initiator must then de-assert P_LOCK# and/or S_LOCK# at the end of the transaction. The PCI 6520 sets the appropriate Status bits, flagging the abnormal Target Termination condition, and normal forwarding of Unlocked Posted and Delayed transactions resumes.

When the PCI 6520 receives a Master or Target Abort in response to a Locked Posted Write transaction, the PCI 6520 ***cannot*** communicate that status to the initiator. The PCI 6520 asserts P_SERR# on the initiator bus when a Master or Target Abort is received during a Locked Posted Write transaction, if the Command register P_SERR# Enable bit is set (PCICR[8]=1; PCI:04h). P_SERR# is asserted for the Master Abort condition if the Bridge Control register Master Abort Mode bit is set (BCNTRL[5]=1; PCI:3Eh).

Note: *The PCI 6520 has an option to ignore the Lock protocol, by clearing the Secondary and/or Primary Lock Enable bits (MSCOPT[14:13]=00b; PCI:46h, respectively).*

14 PCI BUS ARBITRATION

This section describes primary and secondary bus arbitration and bus parking.

14.1 OVERVIEW

The PCI 6520 must arbitrate for use of the secondary bus when forwarding downstream transactions, and for the primary bus when forwarding upstream transactions. The primary bus Arbiter is external to the PCI 6520 (typically located on the motherboard). For the secondary PCI Bus, the PCI 6520 has a built-in Internal Arbiter. The Internal Arbiter can be disabled, allowing use of an External Arbiter for secondary bus arbitration.

14.2 PRIMARY PCI BUS ARBITRATION

The PCI 6520 uses a Request output pin and one Grant input pin (P_REQ# and P_GNT#, respectively) for primary PCI Bus arbitration. The PCI 6520 asserts P_REQ# when forwarding transactions upstream (*that is*, when operating as an initiator on the primary PCI Bus). When there are one or more pending transactions in the upstream direction queues—Posted Write data or Delayed transaction requests—the PCI 6520 maintains P_REQ# assertion. However, if a Target Retry, Disconnect, or Abort is received in response to a PCI 6520-initiated transaction on the primary PCI Bus, the PCI 6520 de-asserts P_REQ# for two PCI Clock cycles. For all cycles passing through the bridge, P_REQ# is not asserted until the complete transaction request is queued.

If the PCI 6520 asserts P_REQ# and the primary bus External Arbiter asserts P_GNT# to grant the bus to the PCI 6520, the PCI 6520 initiates a transaction on the primary bus on the next PCI Clock cycle.

If the primary bus External Arbiter asserts the PCI 6520 P_GNT# signal when P_REQ# is not asserted, the PCI 6520 parks P_AD*x*, P_CBE*x*#, P_PAR, and P_PAR64 by driving these signals to valid logic levels. If the primary bus is parked on the PCI 6520 and the PCI 6520 has a transaction to initiate on the primary bus, the PCI 6520 initiates the transaction if P_GNT# remained asserted during the cycle prior to the start of the transfer.

14.3 SECONDARY PCI BUS ARBITRATION

The PCI 6520 implements a secondary PCI Bus Internal Arbiter, which supports up to eight external bus masters in addition to the PCI 6520. If required, the Internal Arbiter can be disabled, allowing use of an External Arbiter for secondary bus arbitration.

14.3.1 Secondary Bus Arbitration Using Internal Arbiter

To use the Internal Arbiter, the secondary bus Internal Arbiter Enable pin, S_CFN#, must be tied low. The PCI 6520 has eight secondary bus Request input and Grant output pins (S_REQ[7:0]# and S_GNT[7:0]#, respectively) to support external secondary bus masters. If S_CFN# is high, S_REQ0# and S_GNT0# are re-configured as output and input, respectively, and S_GNT[7:1]# and S_REQ[7:1]# are driven high.

The PCI 6520 uses a two-level arbitration scheme, whereby arbitration is divided into two groups—low- and high-priority. The low-priority group represents a single entry in the high-priority group. Therefore, if the high-priority group consists of *n* masters, the highest priority is assigned to the low-priority group at least once every *n*+1 transactions. Priority changes evenly among the low-priority group. Therefore, assuming all masters request the bus, members of the high-priority group are serviced *n* transactions out of *n*+1, while one member of the low-priority group is serviced once every *n*+1 transactions.

Each master can be assigned to a low- or high-priority group, through the Arbiter Control register (ACNTRL; PCI:42h).

Each group can be programmed to use a rotating or fixed-priority scheme, through the Internal Arbiter Control register Group Fixed Arbitration bits (IACNTRL[2, 0]; PCI:50h).

14.3.2 Rotating-Priority Scheme

The secondary Arbiter supports a programmable two-level rotating algorithm that enables the eight request/grant pairs to control up to eight external bus masters. In addition, there is a request/grant pair internal to the PCI 6520, which allows the device to request and be granted access to the secondary bus. Figure 14-1 is an example of the Internal Arbiter wherein four masters, including the PCI 6520, are in the high-priority group, and five masters are in the low-priority group. Using this example, if all requests are always asserted, the highest priority rotates among the masters in the following way (the PCI 6520 is denoted as *B;* high-priority members are provided in *italic* type, and low-priority members in **boldface** type):

B, m0, m1, m2, **m3***, B, m0, m1, m2,* **m4**,
B, m0, m1, m2, **m5**, and so forth

If all masters are assigned to one group, the algorithm defaults to a rotating-priority among all masters. After reset, all external masters are assigned to the low-priority group, and the PCI 6520 to the high-priority group. Therefore, by default, the PCI 6520 receives highest priority on the secondary bus every other transaction and priority rotates evenly among the other masters.

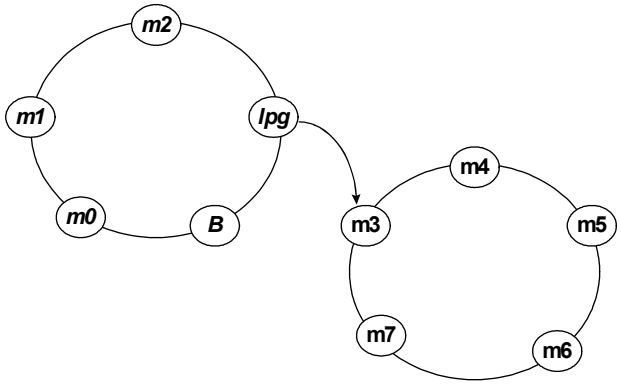


Figure 14-1. Secondary Bus Arbiter Example

Note: *In Figure 14-1, "lpg" denotes "low-priority group."*

Priorities are re-evaluated upon S_FRAME# assertion (*that is*, at the start of each new transaction on the secondary PCI Bus). From this point, until the next transaction starts, the Arbiter asserts the Grant signal corresponding to the highest priority request asserted. If a Grant signal for a particular request is asserted, and a higher priority request subsequently asserts, the Arbiter de-asserts the asserted Grant signal and asserts the Grant signal corresponding to the new higher priority request on the next PCI Clock cycle. When priorities are re-evaluated, 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 within its group. Priority is also re-evaluated if the requesting agent de-asserts its request without generating cycles while the request was granted.

If the PCI 6520 detects that an initiator has failed to assert S_FRAME# after 16 cycles of Grant signal assertion and a secondary bus idle condition, the Arbiter re-evaluates grant assignment. If another initiator asserts REQ# to request the bus, the PCI 6520 switches the grant to the new initiator; otherwise, the same grant is asserted to the same initiator, even if the PCI 6520 does not assert S_FRAME# within 16 cycles.

14.3.3 Fixed-Priority Scheme

The PCI 6520 also supports a fixed-priority scheme within the low- and high-priority groups. In this case, the Internal Arbiter Control register controls whether the low- or high-priority group uses the fixed or rotating-priority scheme (IACNTRL[2, 0]; PCI:50h). If using a fixed-priority scheme, a master within the group is assigned the highest priority within its group, and an option is set to control the priority of other masters relative to the highest priority master. This is controlled through the Internal Arbiter Control register Highest Priority Master and Group Arbitration Order bits (IACNTRL [11:4, 3, 1]; PCI:50h).

Using the example provided in Figure 14-1, but with the groups at fixed-priority, suppose that:

- Master 7 (m7) has the highest priority of the low-priority group (IACNTRL[7:4]=0111b)
- PCI 6520 (B) has the highest priority of the high-priority group (IACNTRL[11:8]=1000b)
- Priority decreases in ascending order of masters for both groups (IACNTRL[3, 1]=00b)

The order of priority with the highest first is as follows:

B, m0, m1, m2, **m7**, **m3**, **m4***,* **m5***,* **m6**

If IACNTRL[3, 1]=11b, priority increases with ascending order of bus master and the order becomes:

B, *m2*, *m1*, *m0*, **m7**, **m6**, **m5**, **m4**, **m3**

Take care when using fixed arbitration in the low-priority group. As previously noted, the low-priority group receives the grant only when there are no high-priority group requests. When the Arbiter switches to the low-priority group, the highest priority master requesting the bus within that group receives the grant. If there are several requests issued by the high-priority group members and the high-priority master in the low-priority group, then lower priority devices in the low-priority group may have to wait before receiving the grant.

To prevent bus contention, if the secondary PCI Bus is idle, the Arbiter waits at least one Clock cycle between the S_GNT*x*# de-assertion and assertion of the next S_REQ*x*#. If the secondary PCI Bus is busy (*that is*, S_FRAME# or S_IRDY# is asserted) when another bus master requests the bus, the Arbiter can de-assert one grant and assert the next grant during the same PCI Clock cycle.

14.3.4 Secondary Bus Arbitration Using External Arbiter

The Internal Arbiter can be disabled by pulling the secondary bus Internal Arbiter Enable pin (S_CFN#) high. An External Arbiter must be used if more than one bus master is required to initiate cycles on the secondary bus.

When S_CFN# is tied high, the PCI 6520 reconfigures two pins to be external Request and Grant pins. S_REQ0# is re-configured to be the external Request output from the PCI 6520 and is used by the PCI 6520 to request the secondary bus. S_GNT0# is re-configured to be the PCI 6520 external Grant input from the External Arbiter.

If the PCI 6520 requests the secondary PCI Bus (S_REQ0# asserted) and the External Arbiter grants the bus to the PCI 6520 (S_GNT0# asserted), the PCI 6520 initiates a transaction on the secondary bus one Clock cycle later.

If the secondary bus External Arbiter asserts S_GNT0# when S_REQ0# is not asserted, the PCI 6520 parks S_AD*x*, S_CBE*x*#, S_PAR, and S_PAR64 by driving these signals to valid logic levels.

When using an External Arbiter, the unused secondary bus Grant outputs (S_GNT[7:1]#) are driven high. Unused secondary bus Request inputs (S_REQ[7:1]#) ***must*** be pulled high.

14.4 ARBITRATION BUS PARKING

Bus parking refers to driving the AD*x*, CBE*x*#, PAR, and PAR64 lines to a known value while the bus is idle. The PCI Bus is parked on the PCI 6520 primary or secondary bus when either or both buses are idle. Bus parking occurs when the bus grant to the PCI 6520 on the parked bus is being asserted, and the PCI 6520 request for that bus is not asserted. The AD*x* and CBE*x*# signals are first driven low (0), then the PAR and PAR64 signals are driven low (0) one cycle later.

When the GNT# signal for the parked bus is de-asserted, the PCI 6520 places the AD*x*, CBE*x*#, PAR, and PAR64 signals into a high-impedance state on the next PCI clock cycle. If the PCI 6520 is parking and wants to initiate a transaction on that bus, the PCI 6520 can start the transaction on the next PCI Clock cycle by asserting FRAME# if GNT# remains asserted.

14.4.1 Software Controlled PCI 64-Bit Extension Signals Parking

By reading the input status of DEV64#, software can determine which PCI 6520 port is interfaced to a backplane and whether it can perform 64-bit transactions.

If only 32-bit transactions can be used, then software can program the PCI 6520 to drive the unused 64-bit extension signals to 0. This is an optional mechanism for use in the event that external pull-up resistors are not desirable (which may be the case in high-speed applications) and prevents the 64-bit extension signals from floating.

The control bits are *Secondary and Primary 64-bit Extension Signals Park*, located in RRC[2:1]; PCI:9Ch, respectively.

15 GPIO INTERFACE

This section describes the GPIO interface pins, control registers, and serial stream.

15.1 GPIO INTERFACE PINS

The PCI 6520 provides eight, general-purpose I/O (GPIO) interface pins. (Refer to Table 15-1.) During normal operation, the Configuration registers control the GPIO interface. During Secondary reset, the GPIO[2:0] and MSK_IN can be used to shift in a 16-bit serial stream that serves as a secondary bus Clock Disable Mask.

The GPIO[7:0] pins have weak internal pull-up resistors. External pull-up or pull-down resistors are recommended.

Table 15-1. GPIO Pin Alternate Functions

GPIO Pin	Alternate Function
GPIO0—Pull-up	Functions as Secondary Bus Clock Mask Shift register clock output when P_RSTIN# is asserted.
GPIO1—Pull-up	***No alternate function.***
GPIO2 —Pull-up	Functions as Shift/Load Control Output to Shift register when P_RSTIN# is asserted.
GPIO3 —Pull-up	***No alternate function.***
GPIO4 —Pull-up	
GPIO5 —Pull-up	
GPIO6 —Pull-up	
GPIO7 —Pull-up	

15.2 GPIO CONTROL REGISTERS

The GPIO registers can be accessed from both sides of the bus. During normal operation, the GPIO interface is controlled by the following three GPIO Configuration registers:

- Output Enable (GPIOOE)
- Output Data (GPIOOD)
- Input Data (GPIOID)

The GPIO[7:4] and GPIO[3:0] Configuration registers consist of five 8-bit fields:

- Output Enable Write 1 to Set (GPIOOE*x*[7:4])
- Output Enable Write 1 to Clear (GPIOOE*x*[3:0])
- Output Data Write 1 to Set (GPIOOD*x*[7:4])
- Output Data Write 1 to Clear (GPIOOD*x*[3:0])
- Input Data (GPIOID*x*[7:4])

The Output Enable fields control whether the GPIO signals are inputs or outputs. Each signal is independently controlled by a bit in each Output Enable field. If 1 is written to the Write 1 to Set field, the corresponding pin is activated as an output. If 1 is written to the Write 1 to Clear field, the output driver is placed into a high-impedance state, and the pin is input only. Writing zeros (0) to these registers has no effect. The reset state for these signals is input only.

The Output Data fields also use the Write 1 to Set and Clear methods. If 1 is written to the Write 1 to Set field and the pin is enabled as an output, the corresponding GPIO output is driven high. If 1 is written to the Write 1 to Clear field and the pin is enabled as an output, the corresponding GPIO output is driven low. Writing zeros (0) to these registers has no effect. The value written to the Output Data register is driven only when the GPIO signal is configured as output. A Type 0 Configuration Write operation is used to program these registers. The reset value for the output is 0.

The Input Data field is Read-Only and reflects the current value of the GPIO[7:0] pins. A Type 0 Configuration Read operation to the Input Data register returns the values of these pins. The GPIO[7:0] pins can be read at any time, whether configured as input only or bi-directional.

15.3 GPIO SERIAL STREAM

Refer to Section 4.2.2, “Secondary Clock Control,” on page 4-1.

16 SUPPORTED COMMANDS

This section discusses the PCI 6520 Conventional PCI and PCI-X command set.

16.1 PRIMARY INTERFACE COMMAND SET

Table 16-1 delineates the PCI 6520 primary interface Conventional PCI and PCI-X command set.

Table 16-1. Primary Interface Supported Commands

P_CBE[3:0]#	Command		Support
	Conventional PCI	PCI-X	
0000b	Interrupt Acknowledge		***Not Supported.***
0001b	Special Cycle		
0010b	I/O Read		If the address is within pass-through I/O range, the transaction is claimed and passed through. If the address points to an I/O-mapped internal bridge register, the transaction is claimed. Otherwise, the transaction is ignored.
0011b	I/O Write		Same as I/O Read (P_CBE[3:0]#=0010b).
0100b	***Reserved***		—
0101b			
0110b	Memory Read	Memory Read DWORD	If the address is within pass-through Memory range, the transaction is claimed and passed through. If the address points to a memory-mapped internal bridge register, the transaction is claimed. Otherwise, the transaction is ignored.
0111b	Memory Write		Same as Memory Read (P_CBE[3:0]#=0110b).
1000b	***Reserved***	Alias to Memory Read Block	**Conventional PCI**—***Not Supported.*** **PCI-X**—Treated as a Memory Read Block (P_CBE[3:0]#=1110b).
1001b	***Reserved***	Alias to Memory Write Block	**Conventional PCI**—***Not Supported.*** **PCI-X**—Treated as a Memory Write Block (P_CBE[3:0]#=1111b).

Table 16-1. Primary Interface Supported Commands (Continued)

P_CBE[3:0]#	Command		Support
	Conventional PCI	PCI-X	
1010b	Configuration Read		Type 0 Configuration Read, claimed if the P_IDSEL line is asserted; otherwise, the read is ignored. If claimed, the target internal register(s) is read. Never passed through. Type 1 Configuration Read, claimed if the P_IDSEL line is asserted; otherwise, the read is ignored. If the target bus is the bridge's secondary bus, the transaction is claimed and passed through as a Type 0 Configuration Read. If the target bus is a subordinate bus that exists behind the bridge (but not equal to the secondary bus), the transaction is claimed and passed through as a Type 1 Configuration Read.
1011b	Configuration Write		Type 0 Configuration Write, same as Configuration Read (P_CBE[3:0]#=1010b). Type 1 Configuration Write (not Special Cycle request), same as Configuration Read (P_CBE[3:0]#=1010b). Configuration Write as Special Cycle request (Device = 1Fh, Function = 7h). If the target bus is the bridge's secondary bus, the transaction is claimed and passed through as a Special Cycle. If the target bus is a subordinate bus that exists behind the bridge (but not equal to the secondary bus), the transaction is claimed and passed through unchanged as a Type 1 Configuration Write.
1100b	Memory Read Multiple	Split Completion	**Conventional PCI**—Treated as a Memory Read (P_CBE[3:0]#=0110b). **PCI-X**—Split Completion.
1101b	DAC		Lower 32 bits of the address are driven out on P_AD[31:0], followed by the upper 32 bits.
1110b	Memory Read Line	Memory Read Block	**Conventional PCI**—Treated as a Memory Read (P_CBE[3:0]#=0110b). **PCI-X**—Memory Read Block.
1111b	Memory Write and Invalidate	Memory Write Block	**Conventional PCI**—Treated as a Memory Write (P_CBE[3:0]#=0111b). **PCI-X**—Memory Write Block.

16.2 SECONDARY INTERFACE COMMAND SET

Table 16-2 delineates the PCI 6520 secondary interface Conventional PCI and PCI-X command set.

Table 16-2. Secondary Interface Supported Commands

S_CBE[3:0]#	Command		Support
	Conventional PCI	PCI-X	
0000b	Interrupt Acknowledge		***Not Supported.***
0001b	Special Cycle		
0010b	I/O Read		If the address is within pass-through I/O range, the transaction is claimed and passed through. If the address points to an I/O-mapped internal bridge register, the transaction is claimed. Otherwise, the transaction is ignored.
0011b	I/O Write		Same as I/O Read (S_CBE[3:0]#=0010b).
0100b	***Reserved***		—
0101b			
0110b	Memory Read	Memory Read DWORD	If the address is within pass-through Memory range, the transaction is claimed and passed through. If the address points to a memory-mapped internal bridge register, the transaction is claimed. Otherwise, the transaction is ignored.
0111b	Memory Write	Memory Write	Same as Memory Read (S_CBE[3:0]#=0110b).
1000b	***Reserved***	Alias to Memory Read Block	**Conventional PCI**—***Not Supported.*** **PCI-X**—Treated as a Memory Read Block (S_CBE[3:0]#=1110b).
1001b	***Reserved***	Alias to Memory Write Block	**Conventional PCI**—***Not Supported.*** **PCI-X**—Treated as a Memory Write Block (S_CBE[3:0]#=1111b).

Table 16-2. Secondary Interface Supported Commands (Continued)

S_CBE[3:0]#	Command		Support
	Conventional PCI	PCI-X	
1010b	Configuration Read		Upstream Configuration Read cycles. ***Not Supported.***
1011b	Configuration Write		Type 0 Configuration Write. ***Not Supported.*** Type 1 Configuration Write (not a Special Cycle request). ***Not Supported.*** Configuration Write as Special Cycle request (Device = 1Fh, Function = 7h). If the target bus is the bridge's primary bus, the transaction is claimed and passed through as a Special Cycle. If the target bus is neither the primary bus nor in the range of buses defined by the bridge's secondary and subordinate bus registers, the transaction is claimed and passed through unchanged as a Type 1 Configuration Write. If the target bus is not the bridge's primary bus, but is within the range of buses defined by the bridge's secondary and subordinate bus registers, the transaction is ignored.
1100b	Memory Read Multiple	Split Completion	**Conventional PCI**—Treated as a Memory Read (S_CBE[3:0]#=0110b). **PCI-X**—Split Completion.
1101b	DAC		Lower 32 bits of the address are driven out on S_AD[31:0], followed by the upper 32 bits.
1110b	Memory Read Line	Memory Read Block	**Conventional PCI**—Treated as a Memory Read (S_CBE[3:0]#=0110b). **PCI-X**—Memory Read Block.
1111b	Memory Write and Invalidate	Memory Write Block	**Conventional PCI**—Treated as a Memory Write (S_CBE[3:0]#=0111b). **PCI-X**—Memory Write Block.

17 BRIDGE BEHAVIOR

This section presents various bridge behavior scenarios that occur when the target responds to a cycle generated by the PCI 6520, on behalf of the initiating master.

17.1 BRIDGE ACTIONS FOR VARIOUS CYCLE TYPES

A PCI cycle is initiated by FRAME# assertion. In a bridge, there are several possibilities for this to occur. Table 17-1 summarizes these possibilities, and delineates the PCI 6520 action for various cycle types.

After the PCI cycle is initiated, a target then has up to four cycles to respond before a Master Abort is initiated. If the target detects an address hit, it asserts DEVSEL# in the cycle corresponding to the Configuration Status register DEVSEL# Timing bits (PCISR[10:9]; PCI:06h or PCISSR[10:9]; PCI:1Eh).

PCI cycle termination can occur in a number of ways. Normal termination begins by the initiator (master) de-asserting FRAME#, with IRDY# being asserted (or remaining asserted) on the same cycle. The cycle completes when TRDY# and IRDY# are simultaneously asserted. The target should de-assert TRDY# for one cycle following final assertion (sustained three-state signal).

Table 17-1. Bridge Actions for Various Cycle Types

Initiator	Target	PCI 6520 Response
Master on primary port	Target on the same primary port	Does not respond. This situation is detected by decoding the address and monitoring P_DEVSEL# for other fast and medium-speed devices on the primary port.
	Target on secondary port	Asserts P_DEVSEL# and normally terminates the cycle if posted; otherwise, returns with a Retry. Next, passes the cycle to the appropriate port. When the cycle completes on the target port, the PCI 6520 waits for the initiator to repeat the same cycle and end with normal termination.
	Target not on primary nor secondary port	Does not respond and the cycle terminates as a Master Abort.
Master on secondary port	Target on the same secondary port	Does not respond.
	Target on primary or other secondary port	Asserts S_DEVSEL# and normally terminates the cycle if posted; otherwise, returns with a Retry. Next, passes the cycle to the appropriate port. When the cycle completes on the target port, the PCI 6520 waits for the initiator to repeat the same cycle and end with normal termination.
	Target not on primary nor other secondary port	Does not respond.

17.2 ABNORMAL TERMINATION (MASTER ABORT, INITIATED BY BRIDGE MASTER)

A Master Abort indicates that the PCI 6520, operating as a master, receives no response from a target (*that is*, no target asserts P_DEVSEL# or S_DEVSEL#). The bridge de-asserts FRAME#, then de-asserts IRDY#.

17.3 PARITY AND ERROR REPORTING

Parity must be checked for all addresses and Write data. Parity is defined on the P_PAR and S_PAR signals. Parity should be even [*that is*, an even number of ones (1)] across AD[63:0], CBE[7:0]#, PAR, and PAR64. Parity information on PAR is valid the cycle after AD[63:0] and CBE[7:0]#, are valid.

For all Address phases, if a Parity error is detected, the error is reported on the P_SERR# signal by asserting P_SERR# for one cycle, then placing two cycles into a high-impedance state after the bad address. P_SERR# can be asserted only if the Command register P_SERR# and Parity Error Response bits are both set to 1 (PCICR[8, 6]=11b; PCI:04h, respectively). For Write Data phases, a Parity error is reported by asserting P_PERR# two cycles after the Data phase and remains asserted for one cycle when PCICR[8]=1. The target reports any type of Data Parity errors during Write cycles, while the master reports Data Parity errors during Read cycles.

Address Parity error detection causes the PCI bridge target to not claim the bus (P_DEVSEL# remains inactive). The cycle then terminates with a Master Abort. When the bridge is operating as master, a Data Parity error during a Read cycle results in the bridge master initiating a Master Abort.

18 PCI FLOW-THROUGH OPTIMIZATION

This section describes Flow-Through optimization, including precautions when using non-optimized PCI master devices, Posted Write and Delayed Read Flow Through, Read cycle optimization, and Read Prefetch boundaries.

18.1 OVERVIEW

The PCI 6520 operates in Flow-Through mode when data from the same transaction is simultaneously transferred on both sides of the bridge (*that is*, data on one side of the bridge "flows through" to the other side of the bridge). The PCI 6520 has several options to optimize PCI transfers after Flow-Though mode is achieved.

Flow-Through mode improves PCI Bus utilization and efficiency. If Data transfers on one side of the bridge are broken into several transactions on the other side of the bridge, poor bus efficiency results. By using Flow-Through mode, the PCI 6520 improves bus efficiency for Posted Writes, Delayed Reads, and reads to Prefetchable space.

18.2 PRECAUTIONS WHEN USING NON-OPTIMIZED PCI MASTER DEVICES

The PCI 6520 is capable of high-performance prefetching. However, some PCI masters may be unable to prefetch large amounts of data. This may be due to a small internal buffer size or other limiting factors. *For example*, if data is being read from a register or FIFO-based architecture, valuable data may be lost if the host prematurely terminates a Prefetch cycle (ideally such spaces would not be listed as prefetchable). Under these circumstances, the default prefetch values may be overly aggressive and affect overall performance. In this case, tune default prefetching by reprogramming the Prefetch registers, as listed in Table 18-1. (Refer to Section 6, "Registers," for a detailed description of these registers.)

The serial EEPROM can also be used to program the Configuration space upon reset.

18.3 POSTED WRITE FLOW THROUGH

During Flow Through of Posted Write cycles, if there is only one Data transfer pending in the Internal Post Memory Write queue, the PCI 6520 de-asserts IRDY# on the target side and waits seven clocks for additional data from the initiator before terminating the cycle. If new Write data is received from the initiator during this period, the PCI 6520 re-asserts IRDY# and continues with the Write cycle. If new Write data is not received during this period, the PCI 6520 terminates the cycle to the target with the last data from the queue and later finishes the cycle.

The Flow-Through Control registers for Posted Writes are detailed in Section 6, "Registers." (Refer to PFTCR[2:0]; PCI:44h and SFTCR[2:0]; PCI:4Eh.)

Table 18-1. Reprogramming Prefetch Registers

Configuration Space Register	Data Value
Primary Initial Prefetch Count (PITLPCNT; PCI:48h)	Same value as Cache Line Size register (PCICLSR; PCI:0Ch). *Most PCs set this value to 08h.*
Secondary Initial Prefetch Count (SITLPCNT; PCI:49h)	
Primary Incremental Prefetch Count (PINCPCNT; PCI:4Ah)	0h
Secondary Incremental Prefetch Count (SINCPCNT; PCI:4Bh)	
Primary Maximum Prefetch Count (PMAXPCNT; PCI:4Ch)	
Secondary Maximum Prefetch Count (SMAXPCNT; PCI:4Dh)	

18.4 DELAYED READ FLOW THROUGH

For Flow Through of Delayed Read cycles, if the Internal Read queue is almost full, the PCI 6520 de-asserts IRDY# on the target side and waits seven clocks for additional data from the initiator before terminating the cycle. If additional space becomes available in the Internal Read queue before the IRDY# inactive period ends, the PCI 6520 re-asserts IRDY# and proceeds with the next Read Data phase. If no additional space becomes available in the Internal Read queue, the current Data phase becomes the last (IRDY# is asserted) and the cycle Disconnects at the end of the Data phase.

The Flow-Through Control registers for Delayed Reads are detailed in Section 6, "Registers." (Refer to PFTCR[6:4]; PCI:44h and SFTCR[6:4]; PCI:4Eh.)

18.5 READ CYCLE OPTIMIZATION

Read Cycle optimization increases the probability of Flow Through occurring during Read accesses to Prefetchable Memory regions. To improve the probability of Flow Through, the amount of data to be prefetched must be correctly configured.

If the PCI 6520 prefetches insufficient data, Flow Through does not occur because prefetching on the target side completes before the Initiator Retries the Read access. Under these circumstances, the Read cycles are divided into multiple cycles.

If the PCI 6520 prefetches excessive data and the internal FIFOs fill, the PCI 6520 must wait for the initiator to Retry the previous Read cycle and then flush the unclaimed data before queuing subsequent cycles.

The initial count is normally equivalent to the cache line size. This assumes that a master usually requires at least one cache line of data. The incremental count is used only when the PCI 6520 does not detect Flow Through for the current cycle being prefetched during the Initial Prefetch Count. The PCI 6520 continues prefetching in increments until it reaches the Maximum Prefetch Count, then Disconnects the cycle.

For Read prefetching, the PCI 6520 implements several registers that control the amount of data prefetched on the primary and secondary PCI Buses. The Prefetch registers listed in Table 18-1 can be used to optimize PCI 6520 performance during Read cycles.

The PCI 6520 prefetches until Flow Through occurs or prefetching must stop, based on the following conditions. Prefetch continues while:

$$(IPMC + IPC + IPC + \ldots + IPC) < MPC$$

where:

IPMC = Initial Prefetch Maximum Count

IPC = Incremental Prefetch Count, < ½ MPC

MPC = Maximum Prefetch Count

If the Prefetch Count did not reach MPC and Flow Through was achieved, the PCI 6520 continues prefetching until the requesting master terminates the Prefetch request. Otherwise, when MPC is reached, the PCI 6520 stops prefetching data.

Incremental Prefetch can be disabled by setting IPC ≥ MPC.

18.5.1 Primary and Secondary Initial Prefetch Count

Assuming that there is sufficient space in the internal FIFO, the Primary and Secondary Initial Prefetch Count registers (PITLPCNT; PCI:48h and SITLPCNT; PCI:49h, respectively) control the amount of data initially prefetched by the PCI 6520 on the primary or secondary bus during reads to the Prefetchable Memory region. If Flow Through is achieved during this initial prefetch, the PCI 6520 continues prefetching beyond this count.

18.5.2 Primary and Secondary Incremental Prefetch Count

The Primary and Secondary Incremental Prefetch Count registers (PINCPCNT; PCI:4Ah and SINCPCNT; PCI:4Bh, respectively) control the amount of prefetching that occurs after the initial prefetch. If Flow Through is not achieved during the initial prefetch, the PCI 6520 attempts to prefetch additional data, until the FIFO fills, or until the Maximum Prefetch Count is reached. Each subsequent prefetch is equal to the Incremental Prefetch Count.

18.5.3 Primary and Secondary Maximum Prefetch Count

The Primary and Secondary Maximum Prefetch Count registers (PMAXPCNT; PCI:4Ch and SMAXPCNT; PCI:4Dh, respectively) limit the amount of prefetched data for a single entry available in the internal FIFO at any time. During Read Prefetch cycles, the PCI 6520 Disconnects the cycle if the data count in the FIFO for the current cycle reaches this value, and Flow Through has not been achieved.

18.6 READ PREFETCH BOUNDARIES

For Memory Read and Memory Read Line commands, the PCI 6520 prefetches from the starting address up to an address with an offset that is a multiple of the Initial Prefetch Count. *For example*, if the starting address is 10h and the Initial Prefetch Count is 20h, the PCI 6520 prefetches only a 10h (20h to 10h) count. After this, the PCI 6520 begins incremental prefetch until the Maximum Prefetch Count is reached, or Flow Through is achieved. The exception to this is in the case of a 64-bit request and six or fewer Dwords from the boundary, or a 32-bit request and four or fewer Dwords from the boundary, in which the PCI 6520 does not activate Incremental Prefetch.

For Memory Read Multiple commands, if the starting address is not 0, the PCI 6520 first prefetches from the starting address up to the address with an offset equal to that of the Initial Prefetch Count. After this, the PCI 6520 prefetches one additional Initial Prefetch Count. *For example*, if the starting address is 10h and the Initial Prefetch Count is 20h, the PCI 6520 first prefetches a 10h (20h to 10h) count, then continues to prefetch another 20h count. Subsequent to this, Incremental Prefetch is invoked until the Maximum Prefetch Count is reached or Flow Through is achieved.

19 FIFO ARCHITECTURE

This section describes FIFO architecture, including how the FIFOs function with Memory Write and Read commands, and how to split the Read FIFO into four 1-KB blocks.

19.1 OVERVIEW

The PCI 6520 contains a 1-KB Write FIFO and 4-KB Read FIFO in both the downstream and upstream directions, for a total of 10 KB of Data FIFO. The FIFO architecture is designed for optimal PCI-X-to-PCI and PCI-X-to-PCI-X bridging, with the following features:

- Flow-Through capable
- Programmable Prefetch Byte Counts of up to 2 KB on the PCI-X port
- Controllable number of outstanding ADQs on the PCI-X port

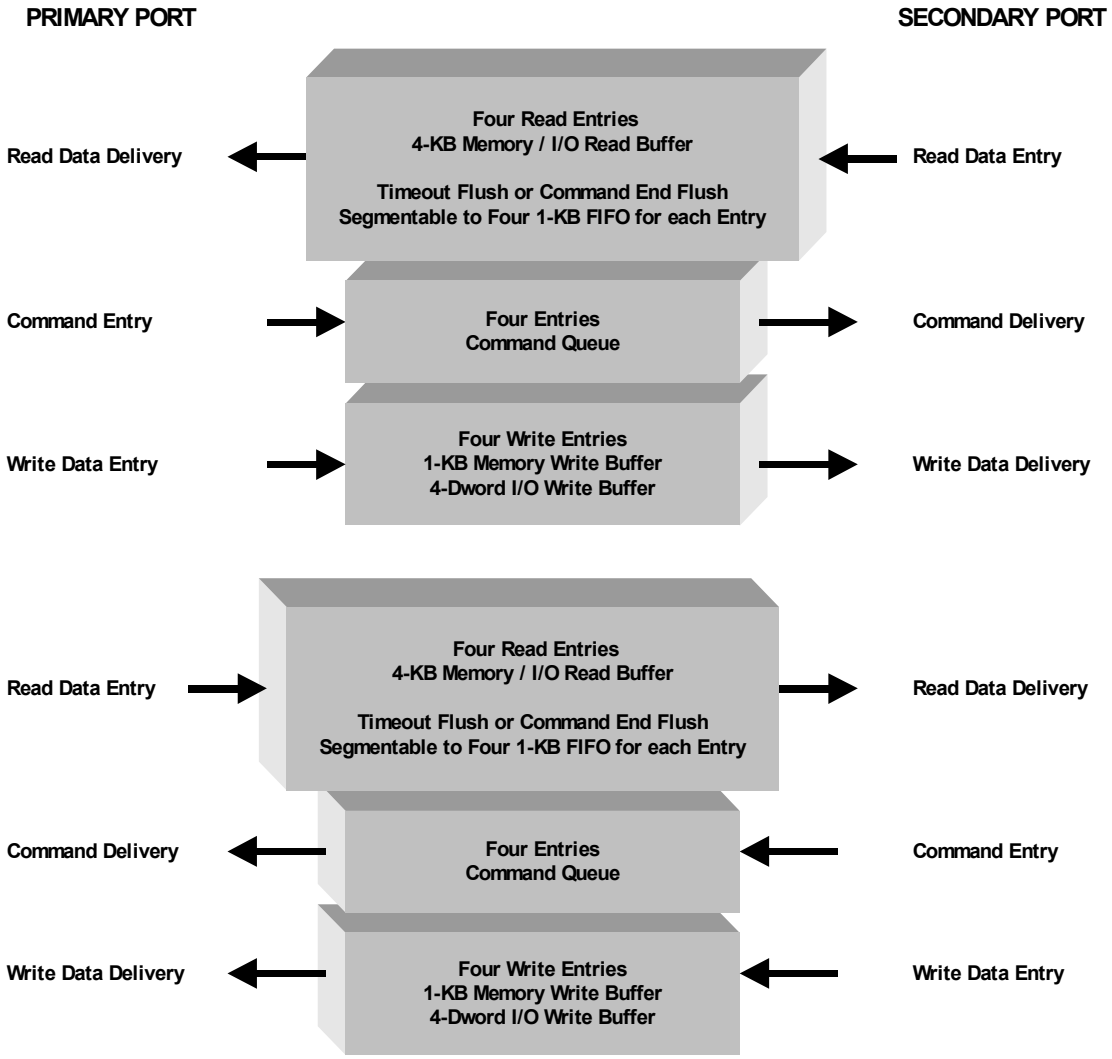


Figure 19-1. PCI 6520 FIFO Architecture

19.2 MEMORY WRITES

This subsection describes PCI-to-PCI-X, PCI-X-to-PCI, and PCI-X-to-PCI-X Memory Writes.

19.2.1 PCI-to-PCI-X Memory Writes

If the initiator writes more than 64 bytes, the PCI 6520 absorbs the greatest amount of data possible into the Write FIFO. When the PCI-X port becomes available and there are 64 or more bytes available in the FIFO, the PCI 6520 begins data delivery from the FIFO to the PCI-X port, using the Byte Count present in the FIFO.

If the initiator Byte Count is less than 64 bytes, the PCI 6520 first receives all the bytes into the FIFO, then delivers the bytes to the PCI-X port.

19.2.2 PCI-X-to-PCI Memory Writes

When functioning as a target and receiving data into the Write FIFO, the PCI 6520 begins data delivery from the FIFO when the opposite port is available. The PCI-X byte count does not influence the start of data delivery to the PCI port.

19.2.3 PCI-X-to-PCI-X Memory Writes

When the PCI 6520 is a target, it always accepts the greatest amount of data possible into the Write FIFO. Concurrent with this process, when the opposite port is available and the incoming data reaches an ADB boundary, the PCI 6520 begins data delivery from the FIFO. If the initiator Byte Count is less than 128 bytes and has not reached an ADB boundary, the PCI 6520 first receives, then delivers, all the bytes.

19.3 MEMORY READS

This subsection describes PCI-to-PCI-X, PCI-X-to-PCI, and PCI-X-to-PCI-X Memory Reads.

19.3.1 PCI-to-PCI-X Memory Reads

When the PCI 6520 receives a PCI Read command, the PCI 6520 issues a Read command to the PCI-X port, using the programmed Prefetch Count specified by the following registers:

- Primary Initial Prefetch Count (PITLPCNT[2:1]; PCI:48h)
- Primary Incremental Prefetch Count (PINCPCNT; PCI:4Ah)
- Secondary Initial Prefetch Count (SITLPCNT[2:1]; PCI:49h)
- Secondary Incremental Prefetch Count (SINCPCNT; PCI:4Bh)

Regardless of the programmed Prefetch Count, the PCI 6520 does not prefetch beyond 16 cache lines. If the count is equal to, or greater than, the maximum programmed Outstanding ADQs Limit (default is 32) in the PCI-X Downstream and Upstream Split Transaction registers (PCIXDNSTR; PCI:FCh and PCIXUPSTR; PCI:F8h, respectively), at least one PCI-X Read command is issued. In general, set the maximum programmed Outstanding ADQs to a value higher than the Prefetch Count so that the PCI 6520 can issue multiple PCI-X Read requests.

When data becomes available in the FIFO, the PCI 6520 begins data delivery to the PCI port.

Prefetched data in the PCI 6520 Read FIFO can either be flushed (if the PCI initiator finishes its current Read transaction) or preserved for a programmed time period. Upon timeout, if the PCI master has not returned to acquire additional data, the FIFO flushes the remaining data. This feature can greatly enhance the PCI-X Bus bandwidth, as the PCI 6520 can prefetch up to 16 cache lines of anticipated data for the PCI devices.

19.3.1.1 Prefetched Data Timeout Flushing

Prefetched data timeout flushing can be controlled by way of the register bits listed in Table 19-1.

Table 19-1. Prefetched Data Timeout Flushing

Control	Description
BUFCR[1]; PCI:4Fh	**Buffer Control Smart Prefetch Enable.** Amount of data prefetched is defined in the Maximum Prefetch Count registers (PMAXPCNT; PCI:4Ch and SMAXPCNT; PCI:4Dh). Defaults to 0. Values after a prefetch command: 0 = Remaining prefetched data is discarded upon completion of the current Read Command. 1 = Remaining prefetched data is ***not*** discarded, but remains available for the next Read Command with consecutive address. The prefetched data is only discarded upon a timeout. The timeout period can be programmed using the Smart Prefetch Timeout bits (BUFCR[6:5]; PCI:4Fh).
BCNTRL[8]; PCI:3Eh	**Bridge Control Primary Master Timeout.** Sets the maximum number of PCI clocks for an initiator on the primary bus to repeat the Delayed Transaction request. Values: 0 = Timeout after 2^{15} PCI clocks 1 = Timeout after 2^{10} PCI clocks Reset to 0.
BCNTRL[9]; PCI:3Eh	**Bridge Control Secondary Master Timeout.** Sets the maximum number of PCI clocks for an initiator on the secondary bus to repeat the Delayed Transaction request. Values: 0 = Timeout after 2^{15} PCI clocks 1 = Timeout after 2^{10} PCI clocks Reset to 0.

19.3.1.2 Setting the Prefetch Count

19.3.1.2.1 PCI Read from Conventional PCI Port

For details on PCI Read from PCI port, refer to Section 18, “PCI Flow-Through Optimization.”

19.3.1.2.2 PCI Read from PCI-X Port

For PCI master reads from PCI-X devices, the PCI 6520 can be set up to prefetch data from the PCI-X port to enhance system performance by minimizing PCI access to the PCI-X Bus.

When the PCI 6520 receives a PCI Read command, the PCI 6520 issues a Read command to the PCI-X port, using the programmed Prefetch Count specified by the following registers:

- Primary Initial Prefetch Count (PITLPCNT[2:1]; PCI:48h)
- Primary Incremental Prefetch Count (PINCPCNT; PCI:4Ah)
- Secondary Initial Prefetch Count (SITLPCNT[2:1]; PCI:49h)
- Secondary Incremental Prefetch Count (SINCPCNT; PCI:4Bh)

19.3.2 PCI-X-to-PCI Memory Reads

After receiving the PCI-X Memory Read command, the PCI 6520 issues a PCI Memory Read command to the PCI port when the port becomes available. When sufficient data is received to read an ADB boundary, the PCI 6520 begins forwarding the Read data to the PCI-X port.

If there is a mid-transaction wait period during which the PCI port is not supplying the PCI 6520 with sufficient data to reach the next ADB boundary, the PCI-X Split Completion cycle is Disconnected so that the PCI 6520 can serve another PCI-X master.

19.3.3 PCI-X-to-PCI-X Memory Reads

Depending on the programmed Outstanding ADQs Limit (default is 32) in the PCI-X Downstream and Upstream Split Transaction registers (PCIXDNSTR; PCI:FCh and PCIXUPSTR; PCI:F8h, respectively)—the PCI 6520 issues a Read command to the opposite port until the requested and pending Read Byte Counts have reached the maximum programmed Outstanding ADQs Limit.

20 POWER MANAGEMENT

This section describes the Power Management feature.

20.1 OVERVIEW

The PCI 6520 incorporates functionality that meets the requirements of *PCI Power Mgmt. r1.1*. These features include:

- PCI Power Management registers, using the Enhanced Capabilities Port (ECP) address mechanism
- Support for D_0, D_{3hot}, and D_{3cold} power management states
- Support for D_0, D_1, D_2, D_{3hot}, and D_{3cold} power management states for devices behind the bridge
- Support for B_2 secondary bus power state when in the D_{3hot} power management state

20.2 POWER MANAGEMENT TRANSITIONS

Table 20-1 delineates the states and related actions the PCI 6520 performs during Power Management transitions. (No other transactions are allowed.)

Table 20-1. States and Related Actions during Power Management Transitions

Current State	Next State	Action
D_0	D_1	During an unimplemented power state, the PCI 6520 ignores the write to the Power State bits (power state remains at D_0, PMCSR[1:0]=00b; PCI:E0h).
	D_2	
	D_{3hot}	If enabled by the BPCC_EN pin, the PCI 6520 disables the secondary clocks and drives them low.
	D_{3cold}	Power removed from the PCI 6520. A power-up reset must be performed to bring the PCI 6520 to D_0.
D_{3hot}	D_0	The PCI 6520 enables secondary clock outputs and performs an internal chip reset. S_RSTOUT# is ***not*** asserted. All registers are returned to the reset values and buffers are cleared.
	D_{3cold}	Power removed from the PCI 6520. A power-up reset must be performed to bring the PCI 6520 to D_0.
D_{3cold}	D_0	During a power-up reset, the PCI 6520 performs the standard power-up reset functions.

21 VPD

This section describes the VPD feature.

The PCI 6520 contains the Vital Product Data (VPD) registers, as specified in *PCI r2.3*. VPD information is stored in the serial EEPROM device, along with Autoload information.

The PCI 6520 provides storage of 192 bytes of VPD data in the serial EEPROM device.

The VPD register block is located at offsets E8h to ECh in PCI Configuration space. (Refer to Section 6.1.2.20, “VPD Capability.”) VPD also uses the Enhanced Capabilities Port Address mechanism.

22 TESTABILITY/DEBUG

This section describes the JTAG interface for use in testing and debugging the PCI 6520.

22.1 JTAG INTERFACE

The PCI 6520 provides a JTAG Boundary Scan interface, which can be utilized to debug a pin's board connectivity.

22.1.1 IEEE 1149.1 Test Access Port

The IEEE 1149.1 Test Access Port (TAP), commonly called the JTAG (Joint Test Action Group) debug port, is an architectural standard described in IEEE Standard 1149.1-1990, *IEEE Standard Test Access Port and Boundary-Scan Architecture.* The standard describes a method for accessing internal chip facilities using a four- or five-signal interface.

The JTAG debug port, originally designed to support scan-based board testing, is enhanced to support the attachment of debug tools. The enhancements, which comply with IEEE Standard 1149.1-1990 specifications for vendor-specific extensions, are compatible with standard JTAG hardware for boundary-scan system testing.

- **JTAG Signals**—JTAG debug port implements the four required JTAG signals—TCK, TDI, TDO, TMS—and the optional TRST# signal. (Refer to Table 3-8, "JTAG/Boundary Scan Pins," on page 3-19 for signal descriptions.)
- **JTAG Clock Requirements**—TCK signal frequency can range from DC to 10 MHz.
- **JTAG Reset Requirements**—JTAG debug port logic and system simultaneously reset. The two methods for placing the PCI 6520 JTAG TAP controller into the Test-Logic-Reset state are as follows:
  - Upon receiving TRST#, the JTAG TAP controller returns to the Test-Logic Reset state
  - Hold the PCI 6520 TMS pin high while transitioning the PCI 6520 TCK pin five times

22.1.2 JTAG Instructions

The JTAG debug port provides the standard **EXTEST**, **SAMPLE/PRELOAD**, **CLAMP**, **HIGHZ**, **IDCODE**, and **BYPASS** instructions. Invalid instructions behave as **BYPASS** instructions.

The PCI 6520 returns the **IDCODE** values listed in Table 22-1. Table 22-2 lists the JTAG instructions, along with their input codes.

Table 22-1. PCI 6520 JTAG IDCODE Value

MSB																										LSB
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	0
4-Bit Version				16-Bit Part Number (PCI 6520 when converted to decimal)																11-Bit PLX Manufacturer Identity											
0	0	1	1	0	0	0	1	1	0	0	1	0	1	1	1	1	0	0	0	0	0	1	1	1	0	1

Table 22-2. JTAG Instructions (IEEE Standard 1149.1-1990)

Instruction	Input Code	Instruction	Input Code
EXTEST	00000b	**HIGHZ**	00101b
SAMPLE/PRELOAD	00001b	**IDCODE**	00110b
CLAMP	00100b	**BYPASS**	11111b

22.1.3 JTAG Boundary Scan

Boundary Scan Description Language (BSDL), IEEE 1149.1b-1994, is a supplement to IEEE Standard 1149.1-1990 and IEEE 1149.1a-1993, *IEEE Standard Test Access Port and Boundary-Scan Architecture*. BSDL, a subset of the IEEE 1076-1993 Standard VHSIC Hardware Description Language (VHDL), allows a rigorous description of testability features in components that comply with the standard. Automated test pattern generation tools use BDSL for package interconnect tests and Electronic Design Automation (EDA) tools for synthesized test logic and verification. BSDL supports robust extensions that can be used for internal test generation and to write software for hardware debug and diagnostics.

The primary components of BSDL include the logical port description, physical pin map, instruction set, and Boundary register description.

The logical port description assigns symbolic names to the PCI 6520 pins. Each pin has a logical type of in, out, in out, buffer, or linkage that defines the logical signal flow direction.

The physical pin map correlates the PCI 6520 logical ports to the physical pins of a specific package. A BSDL description can have several physical pin maps; each map is provided a unique name.

Instruction set statements describe the bit patterns that must be shifted into the Instruction register to place the PCI 6520 in the various Test modes defined by the standard. Instruction set statements also support instruction descriptions unique to the PCI 6520.

The Boundary register description lists each of its cells or shift stages. Each cell has a unique number—the cell numbered 0 is the closest to the Test Data Out (TDO) pin and the cell with the highest number is closest to the Test Data In (TDI) pin. Each cell contains additional information, including:

- Cell type
- Logical port associated with the cell
- Logical function of the cell
- Safe value
- Control cell number
- Disable value
- Result value

22.1.4 JTAG Reset Input TRST#

The TRST# input pin is the asynchronous JTAG logic reset. TRST# assertion causes the PCI 6520 TAP controller to initialize. In addition, when the TAP controller is initialized, it selects the PCI 6520 normal logic path (core-to-I/O). Consider the following when implementing the asynchronous JTAG logic reset on a board:

- If JTAG functionality is required, one of the following should be considered:
  - Use the TRST# input signal low-to-high transition once.
  - Hold the PCI 6520 TMS pin high while transitioning the PCI 6520 TCK pin five times.
- If JTAG functionality is not required, the TRST# signal must be directly connected to ground.

Note: *IEEE Standard 1149.1-1990 requires pull-up resistors on the TDI, TMS, and TRST# pins. To remain PCI r2.3-compliant, no internal pull-up resistors are provided on JTAG pins in the PCI 6520; therefore, the pull-up resistors must be externally added to the PCI 6520 when implementing JTAG.*

23 ELECTRICAL SPECS

This section presents the PCI 6520 electrical specifications.

23.1 GENERAL ELECTRICAL SPECIFICATIONS

The ratings provided in this subsection are those above which the useful life of the PCI 6520 may be impaired.

Use heatsinks when operating in a PCI-X 133 MHz environment. Table 23-1 lists the PCI 6520 maximum ratings. Table 23-2 lists the PCI 6520 functional operating range. Table 23-3 lists the PCI 6520 DC electrical characteristics.

Caution: Stresses greater than the maximums listed in Table 23-1 cause permanent damage to the PCI 6520. This is a stress rating only and functional operation of the PCI 6520 at or above those indicated in the operational sections of this data book is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

Note: *The power consumption for V_{DD_CORE} and V_{DD_IO} is dependent on bus frequency, data traffic, and device loading.*

Table 23-1. Maximum Ratings

Parameter	Minimum	Maximum	Parameter	Minimum	Maximum
Storage Temperature Range	-55 °C	+125 °C	Maximum Voltage to Signal Pins	—	5.5V
Junction Temperature	—	+125 °C	Maximum Power	—	3.0W
V_{DD_IO} Supply Voltage	—	3.9V	Maximum V_{DD_IO} Power (output load dependent)	—	1.2W
V_{DD_CORE} Supply Voltage	—	3.0V	Maximum V_{DD_CORE} Power	—	1.8W
Analog V_{DD} Supply Voltage	—	3.0V	Maximum Analog V_{DD} Power	—	50 mW

Table 23-2. Functional Operating Range

Parameter	Minimum	Maximum	Parameter	Minimum	Maximum
V_{DD_IO} Supply Voltage	3.0V	3.6V	Analog V_{DD} Supply Voltage	1.55V	2.05V
V_{DD_CORE} Supply Voltage	1.55V	2.05V	Operating Ambient Temperature	-40 °C	+85 °C

Table 23-3. DC Electrical Characteristics

Symbol	Parameter	Condition	Minimum	Maximum	Unit	Notes
V_{DD_IO}	V_{DD_IO} Supply Voltage	—	3.0	3.6	V	—
V_{DD_CORE} P_A_{VDD} S_A_{VDD}	V_{DD_CORE} P_A_{VDD} S_A_{VDD}	—	1.55	2.05	V	—
V_{IH}	Input High Voltage	—	0.5 V_{DD_IO}	V_{DD_IO}	V	—
V_{IL}	Input Low Voltage	—	-0.5	+0.3 V_{DD_IO}	V	—
V_{OL}	Output Low Voltage	I_{IOUT}= +1500 µA	—	+0.1 V_{DD_IO}	V	—
V_{OH}	Output High Voltage	I_{IOUT} = -500 µA	0.9 V_{DD_IO}	—	V	—
I_{IL}	Input Leakage Current	$0 < V_{IN} < V_{DD\_IO}$	—	±2	µA	—
C_{IN}	Input Pin Capacitance	—	—	7.0	pF	—

23.2 PLL AND CLOCK JITTER

The PCI 6520 uses two PLLs, one for each interface. These PLLs can be individually disabled by connecting the P_PLLEN# or S_PLLEN# pin to 1.

The minimum input frequency of each PLL is 50 MHz. If a PCI 6520 port is used in a low-speed application (*for example,* at 33 MHz), then disable the appropriate PLL by setting P_PLLEN# or S_PLLEN# to high.

For typical adapter card designs, use the adapter card's M66EN pin to control the PCI 6520's primary PLL by connecting the input of an inverter to the M66EN pin and the output to the PCI 6520's P_PLLEN# input. This ensures that the primary PLL is disabled when operating at 33 MHz. A similar method may be required to control the secondary PLL, depending on the application.

The PCI 6520 uses the PCI-X initialization pattern to automatically enable or disable the primary PLL.

PCI-X r1.0b, Table 6-1 (extrapolated in Table 23-4), defines an add-in card's capabilities according to its PCI-XCAP and M66EN signal states. The primary PLL is enabled or disabled when P_PLLEN# is pulled low, as listed in Table 23-4.

The PLL is sensitive to power and ground noise. A dedicated set of PLL Power and Ground pins are provided to reduce power and ground bounce caused by digital logic feeding into the PLLs. Connect the AV_{DD} pins for each PLL to a clean +1.8V supply and de-couple to the appropriate Ground pins.

Table 23-5 details the PLL operational parameters for the primary and secondary PLLs.

Table 23-4. M66EN and PCI-XCAP Encoding

M66EN	PCI-XCAP	Conventional PCI Device Frequency Capability	PCI-X Device Frequency Capability	Primary PLL Enabled
Ground	Ground	33 MHz	Not Capable	No
	10K-Ohm Pull-Down Resistor		PCI-X 66 MHz	Yes
	Not Connected		PCI-X 133 MHz	
Not Connected	Ground	66 MHz	Not Capable	Yes
	10K-Ohm Pull-Down Resistor		PCI-X 66 MHz	
	Not Connected		PCI-X 133 MHz	

Table 23-5. PLL and Clock Jitter Parameters

Parameter	Minimum	Typical	Maximum	Unit	Condition
Input Frequency	50	—	133	MHz	—
Input Rise and Fall Time	—	—	500	ps	—
Input Cycle-to-Cycle Jitter	-100	—	+100	ps	—
Input Jitter Modulation Frequency	Must be < 100 KHz to allow PLL tracking or > 30 MHz to allow PLL filtering				
Output Cycle-to-Cycle Jitter	-150	—	+150	ps	Clean Power V_{DD} = 1.8V
Output Duty Cycle	45	—	55	%	Clean Power V_{DD} = 1.8V
Phase Lock Time	—	—	100	µs	Clean Power V_{DD} = 1.8V
PLL Power Dissipation	—	9	25	mW	Clean Power V_{DD} = 1.8V $F_{IN} = F_{OUT}$ = 133 MHz
Operating Ambient Temperature	-40	—	+85	°C	—

23.3 PCI/PCI-X SIGNAL TIMING SPECIFICATION

Figure 23-1 illustrates the PCI 6520 signal timing specifications. Table 23-6 delineates the minimum and maximum values, for 66 MHz PCI and 133 MHz PCI-X, for the symbols that appear in Figure 23-1.

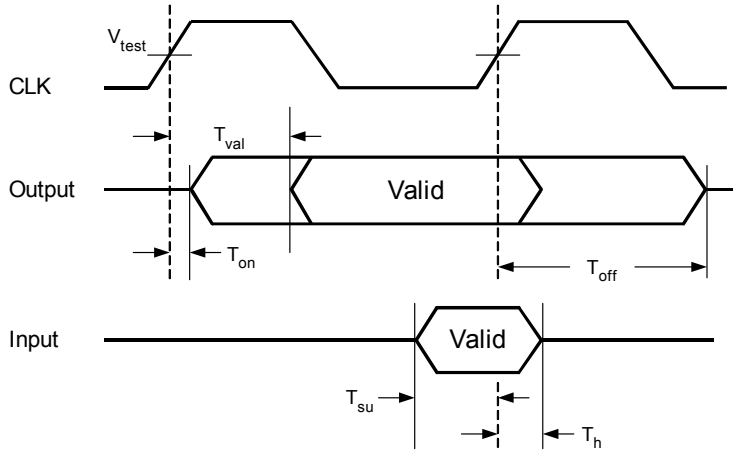


Figure 23-1. PCI/PCI-X Signal Timing Specification

Note: *Refer to PCI-X r1.0b for detailed descriptions of the symbols that appear in Figure 23-1.*

Table 23-6. 66 MHz PCI and 133 MHz PCI-X Signal Timing for Figure 23-1

Symbol	Parameter	66 MHz PCI		133 MHz PCI-X		Unit
		Minimum	Maximum	Minimum	Maximum	
T_{val}	CLK to Signal Valid Delay—Bused Signals	2	6	0.7	3.8	ns
$T_{val(ptp)}$	CLK to Signal Valid Delay—Point to Point	2	6	0.7	3.8	ns
T_{on}	Float to Active Delay	2	—	0	—	ns
T_{off}	Active to Float Delay	—	14	—	7	ns
T_{su}	Input Setup Time to CLK—Bused signals	3	—	1.2	—	ns
$T_{su(ptp)}$	Input Setup Time to CLK—Point to Point	5	—	1.2	—	ns
T_h	Input Signal Hold Time from CLK	0	—	0.5	—	ns
V_{test}	Voltage Test	—	0.4	—	0.4	V_{DD}

24 MECHANICAL SPECS

This section provides the PCI 6520 mechanical dimensions and pinout.

Figure 24-1 illustrates the mechanical dimensions. Table 24-1 lists the mechanical dimensions, in millimeters, unless specified otherwise.

24.1 MECHANICAL DIMENSIONS

The PCI 6520 uses an industry standard 27 x 27 mm 380-pin (ball) PBGA.

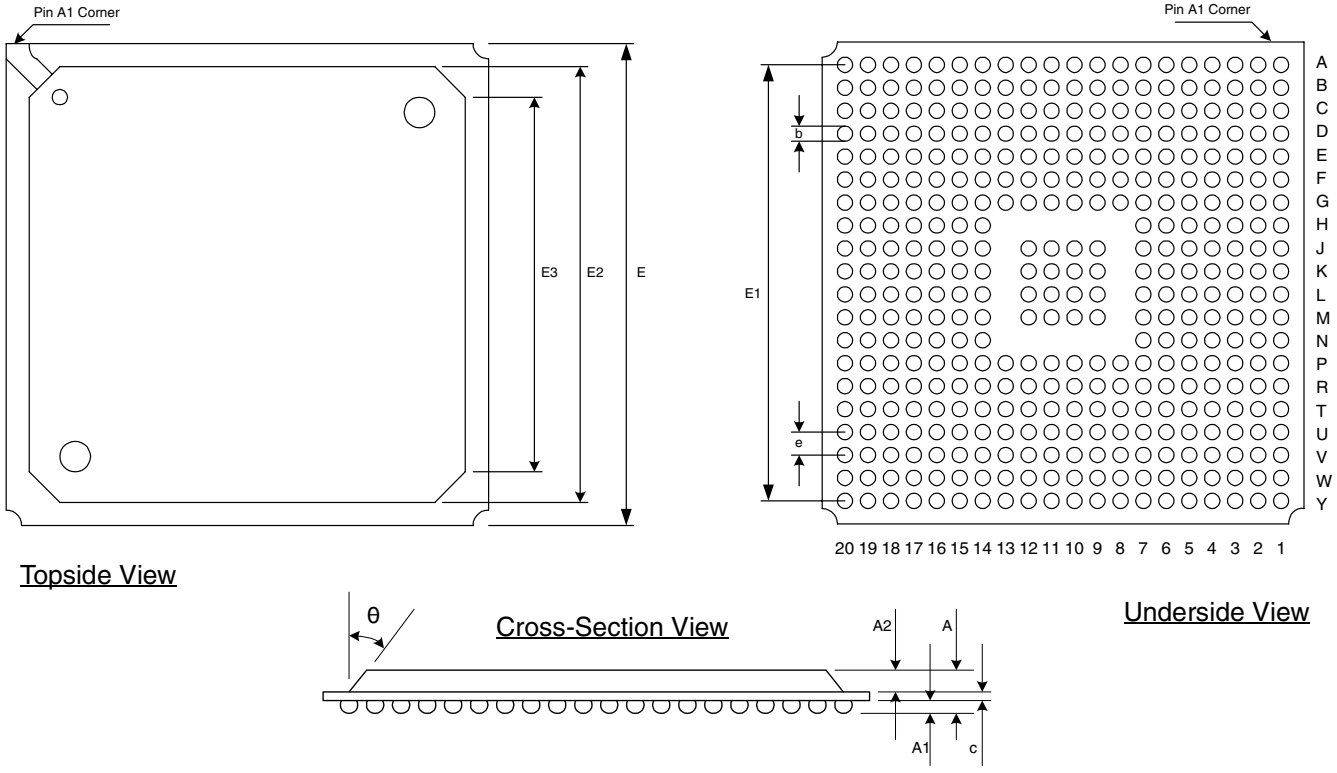


Figure 24-1. PCI 6520 Mechanical Dimensions

Table 24-1. PCI 6520 Mechanical Dimensions for Figure 24-1 Symbols (in Millimeters)

Symbol	Dimension	Minimum	Nominal	Maximum
A	Overall package height	2.20	2.33	2.50
A1	Package standoff height	—	0.60	—
A2	Encapsulation thickness	1.12	1.17	1.22
b	Ball diameter	—	0.75	—
c	Substrate thickness	0.51	0.56	0.61
e	Ball pitch	—	1.27	—
E	Overall package width	26.80	27.00	27.20
E1	—	—	24.13	—
E2	Overall encapsulation width	23.80	24.00	24.20
E3	—	17.95	18.00	18.05
θ	—	—	30°	—

This page intentionally left blank.

24.2 PHYSICAL LAYOUT WITH PINOUT

	1	2	3	4	5	6	7	8	9	10
A	VSS	NC	PCIX100MHZ	NC	GPIO7	S_AD31	S_AD27	VSS	S_AD21	S_AD17
B	S_REQ0#	NC	NC	NC	GPIO6	S_AD30	S_AD26	S_CBE3#	S_AD20	S_AD16
C	S_REQ2#	S_REQ1#	VDD_IO	NC	GPIO5	S_AD29	S_AD25	S_AD23	VSS	S_CBE2#
D	S_REQ5#	S_REQ4#	S_REQ3#	NC	GPIO4	S_AD28	S_AD24	S_AD22	S_AD19	S_FRAME#
E	S_GNT0#	S_VIO	S_REQ7#	S_REQ6#	VSS	S_PLLEN#	NC	VDD_CORE	S_AD18	S_IRDY#
F	S_GNT4#	S_GNT3#	S_GNT2#	S_GNT1#	S_CR	S_AVSS	S_AVDD	VDD_CORE	VDD_IO	VDD_IO
G	P_VIO	S_GNT7#	S_GNT6#	S_GNT5#	S_AVSS	S_AVDD	VDD_CORE	VDD_IO	VSS	VSS
H	S_XCAP_PU	S_RSTOUT#	NC	S_CFN#	VDD_CORE	VDD_CORE	VDD_IO			
J	MSK_IN	BPCC_EN	VSS	PRV_DEV	S_CLKIN	VDD_IO	VSS		VSS	VSS
K	S_CLKO2	S_CLKO1	S_CLKO0	S_CLKOFF	S_CLKIN_STB	VDD_IO	VSS		VSS	VSS
L	NC	P_RSTIN#	VDD_IO	S_CLKO4	S_CLKO3	VDD_IO	VSS		VSS	VSS
M	GPIO0	GPIO1	VSS	GPIO2	GPIO3	VDD_IO	VSS		VSS	VSS
N	OSCSEL#	P_XCAP	PWRGD	P_CLKIN	S_XCAP_IN	VDD_CORE	VDD_IO			
P	P_REQ#	REFCLK	OSCIN	P_GNT#	P_AVSS	P_AVDD	VDD_CORE	VDD_IO	VSS	VSS
R	P_AD29	P_AD30	VDD_IO	P_AD31	P_CLKOE	P_AVSS	P_AVDD	VDD_CORE	VDD_IO	VDD_IO
T	P_AD25	P_AD26	P_AD27	P_AD28	VSS	P_CR	P_PLLEN#	VDD_CORE	P_DEVSEL#	P_SERR#
U	P_AD23	P_IDSEL	P_CBE3#	P_AD24	TMS	VSS	NC	P_CBE2#	P_STOP#	P_PAR
V	P_AD21	P_AD22	VDD_IO	TDI	NC	VDD_IO	P_AD18	P_FRAME#	VSS	P_CBE1#
W	P_AD19	P_AD20	TDO	DEV64#	VSS	NC	P_AD17	P_IRDY#	P_LOCK#	P_AD15
Y	VSS	TCK	TRST#	EEPDATA	EEPCLK	NC	P_AD16	P_TRDY#	P_PERR#	P_AD14
	1	2	3	4	5	6	7	8	9	10

Figure 24-2. PCI 6520 Physical Layout with Pinout—Topside View (A1–A10 through Y1–Y10)

11	12	13	14	15	16	17	18	19	20	
S_TRDY#	S_SERR#	S_AD14	S_AD10	S_AD7	S_AD4	S_AD0	S_CBE7#	S_CBE5#	VSS	A
S_DEVSEL#	S_PAR	S_AD13	S_AD9	S_AD6	S_AD3	S_ACK64#	S_CBE6#	S_CBE4#	S_AD63	B
S_STOP#	VSS	S_AD12	S_AD8	VDD_IO	S_AD2	S_REQ64#	VDD_IO	S_AD61	S_AD62	C
S_LOCK#	S_CBE1#	S_AD11	S_CBE0#	S_AD5	S_AD1	S_AD57	S_AD58	S_AD59	S_AD60	D
S_PERR#	S_AD15	VDD_CORE	S_TST1	S_TST0	VSS	S_AD53	S_AD54	S_AD55	S_AD56	E
VDD_IO	VDD_IO	VDD_CORE	NC	NC	NC	S_AD50	VDD_IO	S_AD51	S_AD52	F
VSS	VSS	VDD_IO	VDD_CORE	NC	S_VIO	S_AD46	S_AD47	S_AD48	S_AD49	G
			VDD_IO	VDD_CORE	VDD_CORE	S_AD42	S_AD43	S_AD44	S_AD45	H
VSS	VSS		VSS	VDD_IO	S_AD38	S_AD39	VSS	S_AD40	S_AD41	J
VSS	VSS		VSS	VDD_IO	S_AD33	S_AD34	S_AD35	S_AD36	S_AD37	K
VSS	VSS		VSS	VDD_IO	S_M66EN	NC	S_CLKRUN#	S_PAR64	S_AD32	L
VSS	VSS		VSS	VDD_IO	P_PAR64	P_CLKRUN#	VSS	NC	P_M66EN	M
			VDD_IO	VDD_CORE	VDD_CORE	P_AD35	P_AD34	P_AD33	P_AD32	N
VSS	VSS	VDD_IO	VDD_CORE	NC	P_VIO	P_AD39	P_AD38	P_AD37	P_AD36	P
VDD_IO	VDD_IO	VDD_CORE	NC	NC	NC	P_AD42	VDD_IO	P_AD41	P_AD40	R
P_AD13	P_AD8	VDD_CORE	P_TST1	P_TST0	VSS	P_AD46	P_AD45	P_AD44	P_AD43	T
P_AD12	P_CBE0#	P_AD5	P_AD1	P_CBE7#	P_CBE4#	P_AD60	P_AD54	P_AD48	P_AD47	U
P_AD11	VSS	P_AD4	P_AD0	VDD_IO	P_AD63	P_AD59	VDD_IO	P_AD51	P_AD49	V
P_AD10	P_AD7	P_AD3	P_ACK64#	P_CBE6#	P_AD62	P_AD58	P_AD55	P_AD52	P_AD50	W
P_AD9	P_AD6	P_AD2	P_REQ64#	P_CBE5#	P_AD61	P_AD57	P_AD56	P_AD53	VSS	Y
11	12	13	14	15	16	17	18	19	20	

Figure 24-3. PCI 6520 Physical Layout with Pinout—Topside View (A11–A20 through Y11–Y20)

A USING PCI 6520

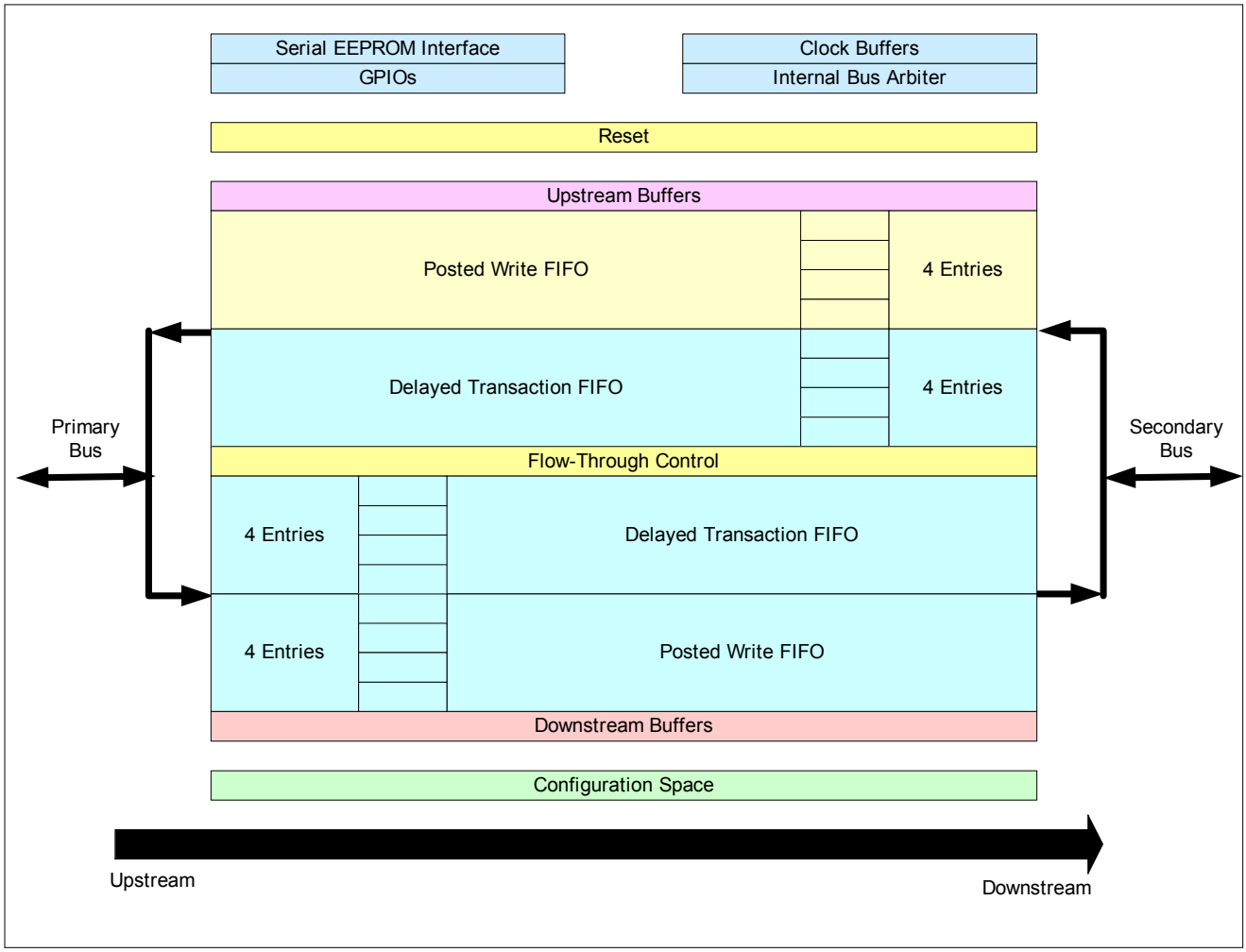


Figure A-1. PCI 6520 Internal Architecture

A.1 APPLICATION

Because the PCI 6520 primary and secondary ports are asynchronous to one another, these two independent systems can run at differing frequencies. The secondary bus can be run faster than the primary bus, and vice versa. The PCI 6520 can be set to enforce PCI-X protocol, without requiring standard PCI-X reset initialization.

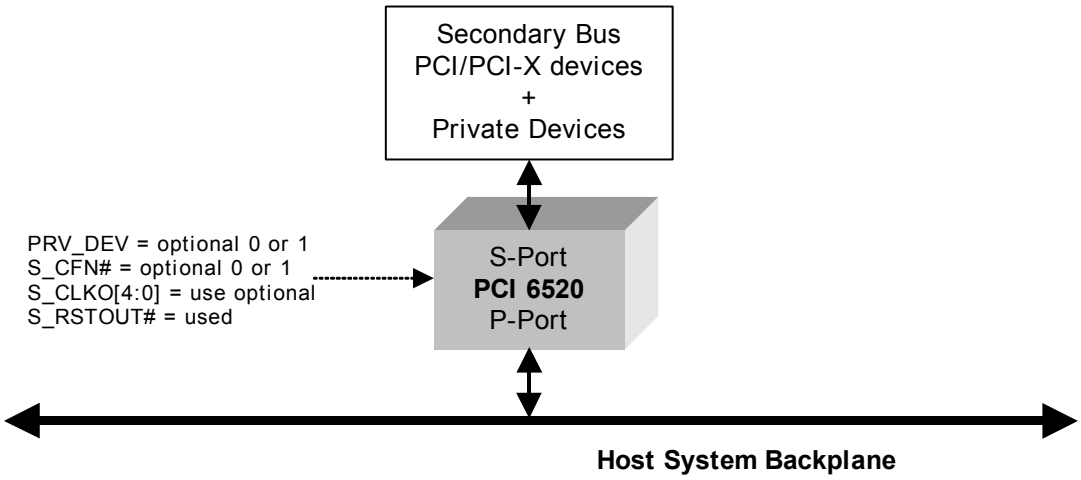
The PCI 6520 controls powerful programmable buffers, which can be used to regulate data throughput for multiple PCI masters on the secondary port. The PCI 6520 can be programmed to prefetch up to 2 KB at a time and the data can be stored in the FIFO.

The host system PCI-X Bus is connected to the PCI 6520 primary port. The secondary PCI port can use a custom-designed External Arbiter or the PCI 6520 Internal Arbiter. To provide clocks to secondary PCI devices and PCI 6520 S_CLKIN, use custom-designed clock generations, PCI 6520 S_CLKO[4:0] outputs (derived out of the primary port PCI clock input), or an external oscillator.

The PCI 6520 also supports private PCI devices on the secondary bus. By setting PRV_DEV to 1, the PCI 6520 allows secondary port IDSEL re-routing, using S_AD[23:16] to S_AD24. When S_AD24 has no device connected to it, S_AD[23:16] Type 1 Access cycles are Master Aborted.

By setting PRV_DEV to 1, and programming the corresponding special Memory Range registers, the PCI 6520 also reserves a Private Memory region for secondary port private device use only. The PCI 6520 does not respond to accesses to this private region by primary or secondary PCI masters.

Figure A-2 provides basic optimization design.



ENFORCE PCI-X: PCI 6520 has a P_XCAP input pin to enforce PCI-X protocol without requiring standard PCI-X reset initialization

Figure A-2. PCI 6520 Basic Optimization Design

B PCI-X CLOCK AND FREQUENCY INITIALIZATION SEQUENCE

B.1 BUS SPEED AND TYPE DETECTION

Figure B-1 provides an example of a Complex Programmable Logic Device (CPLD) being used to detect bus type and speed, and feed to the PCI 6520 during the Reset phase.

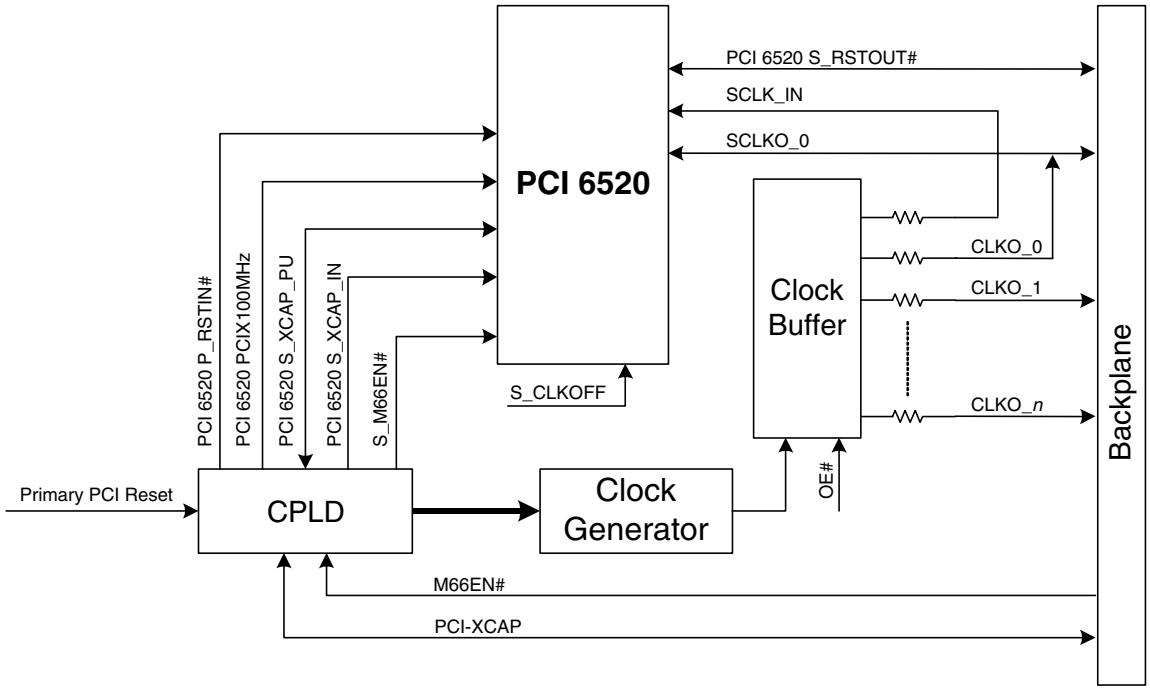


Figure B-1. CPLD Used to Detect Bus Type and Speed and Feed to PCI 6520 during Reset Phase

B.2 SECONDARY CLOCK OUTPUTS

Note: *Secondary clocks from the PCI 6520 are not recommended for PCI-X use. Use high-quality clock buffers for PCI-X applications.*

A high-quality clock generator and clock buffer should be used for 100 and 133 MHz PCI-X applications.

All clocks must have similar flight time, no matter whether they are connected to secondary port PCI-X devices or used for feedback to the PCI 6520 (*that is*, all clock traces are to be of similar length or delay compensated).

B.3 INTERNAL CLOCK DIVIDER

The PCI 6520 internal clock divider works with only with Conventional PCI applications. **Do *not* use the divider for PCI-X applications.**

C PCI 6520CB AND PCI 6540CB PIN COMPARISON

Table C-1. PCI 6520CB Versus PCI 6540CB Pin Assignment Comparison

Pin Location	PCI 6520CB	PCI 6540CB
A2	NC	GPIO15
A3	PCIX100MHz	GPIO13
A4	NC	GPIO10
A8	V_{SS}	S_IDSEL
B2	NC	GPIO14
B3	NC	GPIO12
B4	NC	GPIO9
C4	NC	GPIO8
D4	NC	GPIO11
F15	NC	SLPCIX
H3	NC	S_RSTIN#
L1	NC	P_RSTOUT#
L17	NC	S_PME#
M19	NC	P_PME#
U6	V_{SS}	EJECT
U7	NC	P_BOOT
V5	NC	TRANS#
W5	V_{SS}	U_MODE
W6	NC	L_STAT
Y6	NC	ENUM#

Note: *NC = No Connect pins.*

D GENERAL INFORMATION

The PLX FastLane PCI 6520 is a 64-bit, 133 MHz PCI-X-to-PCI-X bridge that supports transparent operation. The PCI 6520 provides a range of added-value features to system designers, including:

- Two PCI-X ports, each capable of running at the full 64-bit, 133 MHz speed
- Asynchronous primary and secondary ports
- 5V tolerant I/O
- Programmable prefetch
- Programmable Flow Through
- Zero wait state burst
- 10-KB Data FIFO
- Five secondary clock outputs
- Reference clock input for frequency detection
- PCI Power Management support
- Arbitration support for eight secondary bus masters
- Serial EEPROM for configuration
- 8 General Purpose I/O pins
- Vital Product Data (VPD) support
- JTAG boundary scan

The PCI 6520 is offered in an industry-standard 27 x 27 mm 380-pin (ball) PBGA, and is designed to operate over the Industrial Temperature range.

D.1 PACKAGE ORDERING

Table D-1. Available Package

Package	Ordering Part Number
380-pin PBGA	PCI6520-CB13BI

PCI 6520-CB13BI

CB—Part Revision Code
13—Speed Grade (133 MHz PCI-X Bus)
B—Package Type
B = Plastic Ball Grid Array
C—Case Temperature
I = Industrial Temperature
C = Commercial Temperature
ES = Engineering Sample
PCI 6520—Family/Core PCI 6520 device

D.2 UNITED STATES AND INTERNATIONAL REPRESENTATIVES, AND DISTRIBUTORS

A list of PLX Technology, Inc., representatives and distributors can be found at http://www.plxtech.com.

D.3 TECHNICAL SUPPORT

PLX Technology, Inc., technical support information is listed at http://www.plxtech.com/support/, or call 408 774-9060 or 800 759-3735.

INDEX

A

abnormal
  response 8-16
  termination 13-2, 17-2
abort
  master 3-9, 3-11, 3-21, 6-5, 6-10, 6-15, 6-16, 6-32, 6-35, 8-9, 8-11, 8-12, 8-13, 8-15, 8-16, 8-18, 9-1, 9-3, 9-4, 9-10, 9-12, 9-17, 9-18, 9-19, 10-5, 12-13, 13-2, 17-1, 17-2, A-2
  target 3-9, 6-5, 6-10, 6-15, 6-32, 6-35, 8-3, 8-4, 8-7, 8-12, 8-13, 8-14, 8-15, 8-16, 8-18, 9-3, 9-5, 9-9, 9-10, 9-17, 9-18, 9-19, 9-20, 12-1, 12-4, 12-13, 13-2
access, exclusive 13-1–13-2
ACNTRL register 6-17, 6-28, 14-1, 14-2
ADB 9-5–9-22, 19-2
address decoding 10-1–10-5
Allowable Disconnect Boundary
  See ADB
Arbiter Control register 6-2, 6-17, 14-1
arbitration 1-5, 3-11, 3-13, 6-27–6-28, 9-3, 14-1–14-3, D-1
architectural boundary scan
  See IEEE Standard
Attributes 9-1, 9-2, 9-4, 9-5, 9-7–9-8, 9-9, 9-10, 9-12, 9-12–9-14, 9-15, 9-20–9-21
  Relaxed Ordering bit 11-4, 11-6

B

BCNTRL register 4-3, 5-4, 6-4, 6-14–6-15, 6-17, 6-19, 8-8, 10-1, 10-4, 19-3
Boundary Scan
  Description Language 22-2
  pins 3-3, 3-19, 22-1, 22-2
BPCC_EN 3-21, 5-5, 20-1
bridge
  behavior 17-1–17-2
  Control register 6-2, 6-14–6-15
  PCI 6000 series 1-1–1-4
  PCI-X Status register 6-3, 6-52
  Supports Extension register 6-3, 6-48
BSDL
  See Boundary Scan Description Language
BUFCR register 6-27, 6-40, 19-3
Buffer Control register 6-2, 6-27
buffering multiple write transactions 8-5
buffers
  ADB size 9-5
  I/O 22-2
bus operation
  PCI 8-1–8-18
  PCI-X 9-1–9-22

C

CAP_PTR register 6-5, 6-14
CCNTRL register 3-21, 6-16, 6-37, 8-5, 10-5
Chip Control register 3-21, 6-2, 6-16, 6-37, 8-5
CLKCNTRL register 3-3, 3-15, 3-16, 3-17, 4-1, 4-3, 5-3, 6-34
CLKRUN register 6-36
Clock Control register 4-1, 5-3, 6-34
clock, PCI-X B-1–B-2
clocking 4-1–4-6
Clock-Related pins 3-3, 3-15–3-17
commands 16-1–16-4
  conventional PCI-to-PCI-X 9-21
  PCI-X-to-conventional PCI 9-22
  primary 3-7, 6-2
  Primary PCI register 6-4
  read queue 2-2
  secondary 3-11
  serial EEPROM 7-1
completion
  delayed read 8-7–8-8
  delayed write 11-2
  split 9-1–9-20
Control registers 4-1, 5-3, 6-16–6-17, 6-34, 8-6
controller, test access port (TAP)
  See test access port controller

D

DAC 3-6, 3-10, 8-1, 8-2, 9-2, 9-6, 9-10, 9-11, 9-20, 16-2, 16-4
DCNTRL register 5-4, 6-17
deadlock 11-1, 11-2, 11-4–11-6
debug 22-1–22-2
decoding 10-1–10-5
de-coupling, power supply 3-5
delayed read 8-5, 8-7–8-8, 8-16, 13-1, 18-1
delayed read or write 3-9, 6-18, 6-20, 6-26, 6-32, 6-35, 8-2, 8-3, 8-5, 8-12, 8-13, 8-15, 8-17, 8-18, 11-1, 11-2, 11-2–11-3, 12-3, 12-13
DEV64# 3-21, 5-5, 6-29, 6-52
Diagnostic Control register 5-3, 6-17
Dual Address Cycle
  See DAC

E

ECP
20-1
EEPADDR register 6-30, 7-1
EEPCLK 3-19, 5-5, 7-1
EEPCNTRL register 6-29, 7-1
EEPDATA pin 3-19, 5-5, 7-1
EEPDATA register 6-30, 7-1
electrical specs 23-1
Enhanced Capabilities Port
See ECP
error handling 12-1–12-13
exclusive access 13-1–13-2
Extended registers 5-2, 5-7, 6-3, 6-41
EXTRDATA register 5-7, 6-40, 6-41
EXTRIDX register 5-7, 6-40, 6-41, 6-42, 6-43

F

FIFOs 1-5, 6-21, 8-6, 8-7, 11-2, 11-3, 12-1, 12-4, 18-1–18-3, 19-1–19-4, A-2
fixed-priority scheme 14-2–14-3
flow-through 18-1–18-3
primary 6-2, 6-18, 7-3, 18-1, 18-2
secondary 6-2, 6-26, 7-3
FRAME# 17-2

G

GPIO interface 15-1
GPIO[3:0] pins 3-20, 4-1–4-3
GPIO[7:0] pins 3-3, 5-5
GPIO[7:4] pins 3-20
GPIOID[3:0] register 6-33, 15-1
GPIOID[7:4] register 6-39, 15-1
GPIOOD[3:0] register 6-33, 15-1
GPIOOD[7:4] register 6-39, 15-1
GPIOOE[3:0] register 3-20, 6-33, 15-1
GPIOOE[7:4] register 3-20, 6-39, 15-1
Ground pins 3-24, 4-4, 23-3

H

hardware 22-1
Header registers, PCI Type 1 6-4–6-15, 7-3

I

IACNTRL register 6-28, 7-3, 14-1, 14-2, 14-3
IEEE Standard 1149.1-1990 22-1–22-2
IEEE Standard Test Access Port and Boundary-Scan Architecture
See IEEE Standard 1149.1-1990
incremental prefetch count 6-2, 6-24, 7-3, 18-3
initialization 5-1–5-7
interface
debug 22-1–22-2
GPIO 15-1
JTAG 22-1–22-2
primary 12-6, 16-1–16-2
secondary 12-6, 16-3–16-4
Internal Arbiter Control register 6-2, 6-27, 6-28, 7-3, 14-1, 14-2
IRDY# 17-2
ISA 6-14, 10-1, 10-4

J

JTAG 22-1–22-2
pins 3-3, 3-19

L

locks 13-1–13-2

M

M66EN 4-5, 23-3
master abort
See abort, master
mechanical specs 24-1–24-4
memory
prefetchable 6-12–6-13, 10-3
private 6-2, 6-16, 6-37, 10-5, A-2
write and invalidate 6-4, 6-6, 6-21, 8-1, 8-2, 8-3, 8-12, 8-14, 9-21, 11-1, 16-2, 16-4
Miscellaneous Options register 6-2, 6-20–6-21, 7-3, 8-3, 8-9
Miscellaneous pins 3-3, 3-21–3-23
MSCOPT register 3-9, 3-14, 6-20–6-21, 7-3, 8-3, 8-9, 11-2, 13-2
MSK_IN 3-3, 3-15, 4-1–4-3, 5-5

N

NC 3-24
No Connect pins 3-24
normal termination vs. master abort 8-12, 8-13

O

optimization
basic design A-2
flow-through 18-1–18-3
ordering, transaction 11-1–11-6
OSCIN 3-3, 3-15, 4-4, 5-2, 5-5
OSCSEL# 3-3, 3-15, 4-4, 5-5

P

P_ACK64# 3-2, 3-6, 5-5
P_AD[31:0] 3-6, 16-2
P_AD[63:0] 3-2, 5-5
P_AD[63:32]
6-38

P_AV_{DD}
3-24, 4-4, 5-5, 23-3
P_AV_{SS} 3-24, 5-5
P_CBE[3:0]# 3-6, 8-2, 8-9, 16-1–16-2
P_CBE[7:0]# 3-2, 5-5
P_CBE[7:4]# 3-7, 6-38, 8-9
P_CLKIN 3-3, 3-15, 4-1, 4-4, 4-6, 5-2, 5-3, 5-5, 6-31
P_CLKOE 3-3, 3-15, 5-6, 6-29
P_CLKRUN# 3-21, 5-6
P_CR 3-3, 3-15, 5-6
P_DEVSEL# 3-2, 3-7, 5-6, 8-8, 12-1, 17-1, 17-2
P_FRAME# 3-2, 3-7, 5-6, 13-2
P_GNT# 3-2, 3-7, 5-6, 14-1
P_IDSEL 3-2, 3-7, 5-6, 8-8, 16-2
P_IRDY# 3-2, 3-7, 5-6, 6-26
P_LOCK# 3-2, 3-7, 5-6, 13-1–13-2
P_M66EN 3-2, 3-8, 3-15, 4-4, 5-2, 5-6
P_PAR 3-2, 3-8, 5-6, 14-1
P_PAR64 3-2, 3-8, 5-6, 6-38, 14-1, 17-2
P_PERR# 3-2, 3-8, 5-6, 12-1–12-13
P_PLLEN# 3-3, 3-15, 4-4, 5-6, 23-3
P_REQ# 3-2, 3-8, 5-6, 14-1
P_REQ64# 3-2, 3-9, 5-6
P_RSTIN# 3-18, 4-1, 5-2, 5-4, 5-5–5-6, 7-1, 9-1
P_SERR# 3-2, 3-9, 5-6, 6-2, 6-5, 6-15, 6-32, 6-35, 8-4, 8-7, 8-8, 8-13, 8-14, 8-15, 8-16, 12-1–12-13, 13-2, 17-2
P_STOP# 3-2, 3-9, 5-6
P_TRDY# 3-2, 3-10, 5-6
P_TST[1:0] 3-21, 5-6
P_V_{IO} 3-4, 3-24, 5-5
P_XCAP 3-22, 5-1, 5-6
package specs 24-1–24-4
parity 12-1–12-13
  error rules 9-4
  primary signal 3-8
  reporting errors 17-2
  secondary signal 3-12, 3-13
PBGA
  industry standard 24-1
  package ordering information D-1
  pinout 24-4–24-5
PCI 6520
  general product information 1-1–1-6, D-1
PCI arbitration 14-1–14-3
PCI Bus operation 8-1–8-18
PCI Bus Power Management Interface Specification, Revision 1.1
  See PCI Power Mgmt. r1.1
PCI Configuration registers 6-2–6-53
PCI Local Bus Specification, Revision 2.3
  See PCI r2.3
PCI Power Mgmt. r1.1 2-1, 20-1
PCI r2.3 1-5, 3-2, 5-2, 11-3, 21-1, 22-2
PCI to PCI Bridge Architecture Specification, Revision 1.2
  See P-to-P Bridge r1.2
PCI transactions 8-1–8-18, 11-1–11-3
PCI Type 1 Header registers 6-4–6-15
PCIBISTR register 6-7, 7-3
PCICCR register 6-6, 6-38
PCICLSR register 6-6, 6-22, 6-23, 8-3, 18-1
PCICR register 6-4–6-5, 6-10, 6-14, 6-15, 6-32, 7-3, 8-4, 8-7, 8-13, 8-14, 8-15, 8-16, 10-1, 10-2, 10-4, 10-5, 12-1, 12-4, 12-6–12-13, 13-2, 17-2
PCIHTR register 6-7, 6-38, 7-3
PCIIDR register 6-4, 6-38, 7-3
PCIIOBAR register 6-9, 6-14, 10-1, 10-2
PCIIOBARU16 register 6-9, 6-13, 10-2
PCIIOLMT register 6-9, 10-2
PCIIOLMTU16 register 6-9, 6-13, 10-2
PCIIPR register 6-14
PCILTR register 6-6
PCIMBAR register 6-11, 10-2, 10-3
PCIMLMT register 6-11, 10-2, 10-3
PCIPBNO register 6-8, 8-11
PCIPMBAR register 6-12, 6-13, 10-2, 10-4
PCIPMBARU32 register 6-12, 6-13, 10-2
PCIPMLMT register 6-12, 6-13, 10-2, 10-4
PCIPMLMTU32 register 6-12, 6-13, 10-2
PCIREV register 6-6
PCISBNO register 6-8, 8-9, 8-11
PCISLTR register 6-8
PCISR register 6-5, 8-13, 8-14, 8-15, 8-16, 12-1, 12-2, 12-4, 12-6, 12-8, 12-13, 17-1
PCISSR register 6-10, 8-13, 8-14, 8-15, 8-16, 12-1, 12-4, 12-7, 12-9, 12-13, 17-1
PCISUBNO register 6-8, 8-11
PCI-X Addendum to PCI Local Bus Specification, Revision 1.0b
  See PCI-X r1.0b
PCI-X Bus operation 9-1–9-22
PCI-X Capability registers 6-50–6-53
PCI-X clock B-1–B-2
PCI-X r1.0b 3-2, 3-3, 6-1, 11-4, 23-5
PCI-X transactions 9-1–9-22, 11-4–11-6
PCIX_NEXT register 6-50
PCIX100MHZ 3-20, 5-1, 5-5
PCIXBSR register
  3-21, 6-52, 9-8, 9-12, 9-14

PCIXCAPID register
6-50
PCIXDNSTR register 6-51, 6-53, 19-2
PCIXSSR register 3-20, 6-50–6-51
PCIXUPSTR register 6-52, 6-53, 19-4
PFTCR register 6-18, 7-3, 18-1, 18-2
Philips 74F166 4-2–4-3
physical specs 24-1–24-4
pin states, during PWRGD and primary reset 5-5–5-6
PINCPCNT register 6-22, 6-24, 7-3, 18-1, 18-3, 19-2, 19-4
pinout 3-6–3-24
PBGA 24-4–24-5
specs 24-1–24-4
pins
Boundary Scan 3-3, 3-19, 22-1, 22-2
BPCC_EN 3-21, 5-5, 20-1
Clock Related 3-3, 3-15–3-17
DEV64# 3-21, 5-5, 6-29, 6-52
EEPCLK 3-19, 5-5, 7-1
EEPDATA 3-19, 5-5, 7-1
FRAME# 17-2
GPIO[3:0] 3-20, 4-1–4-3
GPIO[7:0] 3-3, 5-5
GPIO[7:4] 3-20
Ground 3-24, 4-4, 23-3
IRDY# 17-2
JTAG 3-3, 3-19, 22-1
M66EN 4-5, 23-3
Miscellaneous 3-3, 3-21–3-23
MSK_IN 3-3, 3-15, 4-1–4-3, 5-5
NC 3-24
No Connect 3-24
OSCIN 3-3, 3-15, 4-4, 5-2, 5-5
OSCSEL# 3-3, 3-15, 4-4, 5-5
P_ACK64# 3-2, 3-6, 5-5
P_AD[31:0] 3-6, 16-2
P_AD[63:0] 3-2, 5-5
P_AD[63:32] 6-38
P_AV_{DD} 3-24, 4-4, 5-5, 23-3
P_AV_{SS} 3-24, 5-5
P_CBE[3:0]# 3-6, 8-2, 8-9, 16-1–16-2
P_CBE[7:0]# 3-2, 5-5
P_CBE[7:4]# 3-7, 6-38, 8-9
P_CLKIN 3-3, 3-15, 4-1, 4-4, 4-6, 5-2, 5-3, 5-5, 6-31
P_CLKOE 3-3, 3-15, 5-6, 6-29
P_CLKRUN# 3-21, 5-6
P_CR 3-3, 3-15, 5-6
P_DEVSEL# 3-2, 3-7, 5-6, 8-8, 12-1, 17-1, 17-2
P_FRAME# 3-2, 3-7, 5-6, 13-2
P_GNT# 3-2, 3-7, 5-6, 14-1
P_IDSEL 3-2, 3-7, 5-6, 8-8, 16-2
P_IRDY# 3-2, 3-7, 5-6, 6-26
P_LOCK# 3-2, 3-7, 5-6, 13-1–13-2
P_M66EN 3-2, 3-8, 3-15, 4-4, 5-2, 5-6
P_PAR 3-2, 3-8, 5-6, 14-1
P_PAR64 3-2, 3-8, 5-6, 6-38, 14-1, 17-2
P_PERR# 3-2, 3-8, 5-6, 12-1–12-13
P_PLLEN# 3-3, 3-15, 4-4, 5-6, 23-3
P_REQ# 3-2, 3-8, 5-6, 14-1
P_REQ64# 3-2, 3-9, 5-6
P_RSTIN# 3-18, 3-20, 4-1, 5-2, 5-4, 5-5–5-6, 7-1, 9-1, 15-1
P_SERR# 3-2, 3-9, 5-6, 6-2, 6-5, 6-15, 6-32, 6-35, 8-4, 8-7, 8-8, 8-13, 8-14, 8-15, 8-16, 12-1–12-13, 13-2, 17-2
P_STOP# 3-2, 3-9, 5-6
P_TRDY# 3-2, 3-10, 5-6
P_TST[1:0] 3-21, 5-6
P_V_{IO} 3-4, 3-24, 5-5
P_XCAP 3-22, 5-1, 5-6
PCIX100MHZ 3-20, 5-1, 5-5
Power 3-24, 4-4, 23-3
Primary Clock 3-3, 3-15, 3-21, 4-1, 4-6, 5-2, 20-1
Primary PCI Bus Interface 3-2, 3-6–3-10
PRV_DEV 3-21, 5-6, 6-16, 6-37, 10-5, A-2
PWRGD 3-18, 5-2–5-6, 5-7
REFCLK 3-3, 3-15, 4-6, 5-5
Reset 3-3, 3-18
S_ACK64# 3-2, 3-10, 5-5
S_AD[31:0] 3-10, 16-4
S_AD[63:0] 3-2, 5-5
S_AD[63:32] 3-10, 6-38
S_AV_{DD} 3-24, 4-4, 5-5, 23-3
S_AV_{SS} 3-24, 5-5
S_CBE[3:0]# 3-11, 16-3–16-4
S_CBE[7:0]# 3-2, 5-5
S_CBE[7:4]# 3-11, 6-38
S_CFN# 3-22, 5-5, 6-29, 14-1
S_CLKIN 3-3, 3-16, 4-1, 4-3, 4-4, 4-6, 5-5, 6-29, 6-31, A-2
S_CLKIN_STB 3-3, 3-16, 4-4, 5-5
S_CLKO[4:0] 4-1–4-4, 5-5, 6-48
S_CLKO[4:1] 3-3, 3-16, 3-17, 4-1
S_CLKO0 3-3, 3-16, 3-17, 6-34, B-2
S_CLKOFF 3-3, 3-17, 4-1, 5-5
S_CLKRUN# 3-22, 5-5, 6-36
S_CR 3-3, 3-17, 5-5
S_DEVSEL# 3-2, 3-11, 5-3, 5-5, 12-1, 17-1, 17-2
S_FRAME# 3-2, 3-11, 5-5, 13-2, 14-2, 14-3
S_GNT[7:0]# 3-2, 5-6, 14-1
S_GNT[7:1]# 3-12, 14-1
S_GNT0# 3-11, 14-1, 14-3
S_IDSEL 10-5
S_IRDY# 3-2, 3-12, 5-6, 6-18, 14-3
S_LOCK# 3-2, 3-12, 5-6, 13-1–13-2
S_M66EN 3-2, 3-12, 4-4, 5-1, 5-2, 5-6

S_PAR 3-2, 3-12, 5-6, 14-3
S_PAR64 3-2, 3-13, 5-6, 6-38, 14-3, 17-2
S_PERR# 3-2, 3-13, 5-6, 6-10, 12-1–12-13
S_PLLEN# 3-17, 4-4, 5-6, 23-3
S_REQ[7:0]# 3-2, 5-6, 6-17, 14-1
S_REQ[7:1]# 3-13, 14-1
S_REQ0# 3-13, 14-1, 14-3
S_REQ64# 3-2, 3-14, 5-6
S_RSTOUT# 3-18, 4-3, 4-4, 5-2, 5-3, 5-4, 5-6, 6-15, 6-17, 6-51, 20-1
S_SERR# 3-2, 3-14, 6-14, 12-1, 12-5, 12-12–12-13
S_STOP# 3-2, 3-14, 5-3, 5-6
S_TRDY# 3-2, 3-14, 5-3, 5-6
S_TST[1:0] 3-22, 5-6
S_V_{IO} 3-4, 3-24, 5-6
S_XCAP_IN 3-23, 5-1, 5-6, 9-1
S_XCAP_PU 3-23, 5-1, 5-6
Secondary Clock 3-3, 3-16, 3-17, 3-21, 3-22, 4-1, 4-4, 4-6, 5-2, 20-1
Secondary PCI Bus Interface 3-2, 3-10–3-14
Serial EEPROM 3-3, 3-19
TCK 3-19, 22-1
TDI 3-19, 22-1, 22-2
TDO 3-19, 22-1, 22-2
TMS 3-19, 22-1
TRST# 3-19, 22-1, 22-2
V_{DD_CORE} 3-24, 5-6
V_{DD_IO} 3-24, 5-6
V_{SS} 3-15, 3-17, 3-24, 5-6
See Also pinout designations and Appendix C, *PCI 6520CB and PCI 6540CB Pin Comparison*
PITLPCNT register
6-22, 6-24, 7-3, 18-1, 18-2, 19-2, 19-4
PLX Technology, Inc.
product information 1-1
product ordering and technical support D-2
PMAXPCNT register 6-25, 6-27, 7-3, 18-1, 18-3, 19-3
PMC register 6-38, 6-46, 6-47, 7-3
PMCAPID register 6-46
PMCDATA register 6-38, 6-48, 7-3
PMCSR register 5-4, 6-3, 6-38, 6-48, 7-3, 20-1
PMCSR_BSE register 6-48
PMNEXT register 6-46
power dissipation 1-2–1-3, 3-4, 4-1, 4-5, 23-1, 23-4
power good
input signal 3-18
reset 5-2–5-3
See Also PWRGD
Power Management 20-1
Power Management Capability registers 6-3, 6-46–6-48, 7-3
Power pins 3-24, 4-4, 23-3
power supply 3-5
prefetch 18-3
data timeout flushing 19-3
incremental count 6-2, 6-24, 7-3
memory 6-12–6-13, 10-3
read transaction 8-5–8-6
reprogramming registers 18-1
setting byte count 19-4
smart 6-27, 6-40–6-45, 19-3
Prefetch Control registers 6-2, 6-22–6-25, 7-3, 18-1, 18-2, 19-2, 19-4
Primary
clock frequency measurement 4-6
Clock pins 3-3, 3-15, 3-21, 4-1, 4-6, 5-2, 20-1
Flow-Through Control registers 6-18
PCI Bus Interface pins 3-2, 3-6–3-10
priority schemes 14-2–14-3
private memory 6-2, 6-16, 6-37, 10-5, A-2
PRV_DEV 3-21, 5-6, 6-16, 6-37, 10-5, A-2
PSERRED register 6-32, 8-13–8-16, 12-1, 12-3
PSERRSR register 6-35, 12-1
P-to-P Bridge r1.2 6-1, 10-3, 12-13
pull-up/pull-down resistor recommendations 3-2–3-4
PVPD_NEXT register 6-49
PVPDAD register 6-49
PVPDATA register 6-49
PVPDID register 6-49
PVTMBAR register 3-21, 6-16, 6-37
PVTMBARU32 register 3-21, 6-37
PVTMLMT register 3-21, 6-37
PVTMLMTU32 register 6-37
PWRGD 3-18, 5-2–5-6, 5-7

R

Read-Only Control register 6-38
REFCLK 3-3, 3-15, 4-6, 5-5
registers
ACNTRL 6-17, 6-28, 14-1, 14-2
Arbiter Control 6-2, 6-17, 14-1
BCNTRL 4-3, 5-4, 6-4, 6-14–6-15, 6-17, 6-19, 8-8, 10-1, 10-4, 19-3
BUFCR 6-27, 6-40, 19-3
Buffer Control 6-2, 6-27
CAP_PTR 6-5, 6-14
CCNTRL 3-21, 6-16, 6-37, 8-5, 10-5
Chip Control 3-21, 6-2, 6-16, 6-37, 8-5
CLKCNTRL 3-3, 3-15, 3-16, 3-17, 4-1, 4-3, 5-3, 6-34
CLKRUN 6-36
Clock Control 3-15, 3-16, 4-1, 4-3, 5-3, 6-34
Control 3-15, 3-16, 4-1, 4-3, 5-3, 6-16–6-17, 6-28, 6-34, 6-37, 8-6
DCNTRL 5-4, 6-17
Diagnostic Control 5-3, 6-17

EEPADDR 6-30, 7-1
EEPCNTRL 6-29, 7-1
EEPDATA 6-30, 7-1
Extended 5-2, 5-7, 6-3, 6-41
EXTRDATA 5-7, 6-40, 6-41
EXTRIDX 5-7, 6-41, 6-42, 6-43
GPIOID[3:0] 6-33, 15-1
GPIOID[7:4] 6-39, 15-1
GPIOOD[3:0] 6-33, 15-1
GPIOOD[7:4] 6-39, 15-1
GPIOOE[3:0] 3-20, 6-33, 15-1
GPIOOE[7:4] 3-20, 6-39, 15-1
Header, PCI Type 1 6-4–6-15, 7-3
IACNTRL 6-28, 7-3, 14-1, 14-2, 14-3
Internal Arbiter Control 6-2, 6-28, 7-3, 14-1, 14-2
Miscellaneous Options 6-2, 6-20–6-21, 7-3, 8-3
MSCOPT 3-9, 3-14, 6-20–6-21, 7-3, 8-3, 8-9, 11-2, 13-2
PCI Configuration 6-2–6-53
PCI Type 1 Header 6-4–6-15
PCIBISTR 6-7, 7-3
PCICCR 6-6, 6-38
PCICLSR 6-6, 6-22, 6-23, 8-3, 18-1
PCICR 6-4–6-5, 6-10, 6-14, 6-15, 6-32, 7-3, 8-4, 8-7, 8-13, 8-14, 8-15, 8-16, 10-1, 10-2, 10-4, 10-5, 12-1, 12-4, 12-6–12-13, 13-2, 17-2
PCIHTR 6-7, 6-38, 7-3
PCIIDR 6-4, 6-38, 7-3
PCIIOBAR 6-9, 6-14, 10-1, 10-2
PCIIOBARU16 6-9, 6-13, 10-2
PCIIOLMT 6-9, 10-2
PCIIOLMTU16 6-9, 6-13, 10-2
PCIIPR 6-14
PCILTR 6-6
PCIMBAR 6-11, 10-2, 10-3
PCIMLMT 6-11, 10-2, 10-3
PCIPBNO 6-8, 8-11
PCIPMBAR 6-12, 6-13, 10-2, 10-4
PCIPMBARU32 6-12, 6-13, 10-2
PCIPMLMT 6-12, 6-13, 10-2, 10-4
PCIPMLMTU32 6-12, 6-13, 10-2
PCIREV 6-6
PCISBNO 6-8, 8-9, 8-11, 9-12
PCISLTR 6-8
PCISR 6-5, 8-13, 8-14, 8-15, 8-16, 12-1, 12-2, 12-4, 12-6, 12-8, 12-13, 17-1
PCISSR 6-10, 8-13, 8-14, 8-15, 8-16, 12-1, 12-4, 12-7, 12-9, 12-13, 17-1
PCISUBNO 6-8, 8-11
PCIX_NEXT 6-50
PCIXBSR 3-21, 6-52, 9-8, 9-12, 9-14
PCIXCAPID 6-50
PCIXDNSTR 6-51, 6-53, 19-2
PCIXSSR 3-20, 6-50–6-51
PCIXUPSTR 6-52, 6-53, 19-4
PFTCR 6-18, 7-3, 18-1, 18-2
PINCPCNT 6-22, 6-24, 7-3, 18-1, 18-3, 19-2, 19-4
PITLPCNT 6-22, 6-24, 7-3, 18-1, 18-2, 19-2, 19-4
PMAXPCNT 6-25, 6-27, 7-3, 18-1, 18-3, 19-3
PMC 6-38, 6-46, 6-47, 7-3
PMCAPID 6-46
PMCDATA 6-38, 6-48, 7-3
PMCSR 5-4, 6-3, 6-38, 6-48, 7-3, 20-1
PMCSR_BSE 6-48
PMNEXT 6-46
Power Management Capability 6-3, 6-46–6-48, 7-3
Prefetch Control 6-22–6-25, 7-3, 18-1, 18-2, 19-2, 19-4
Primary Flow-Through Control 6-18
PSERRED 6-32, 8-13–8-16, 12-1, 12-3
PSERRSR 6-35, 12-1
PVPD_NEXT 6-49
PVPDAD 6-49
PVPDATA 6-49
PVPDID 6-49
PVTMBAR 3-21, 6-16, 6-37
PVTMBARU32 3-21, 6-37
PVTMLMT 3-21, 6-37
PVTMLMTU32 6-37
Read-Only Control 6-38
RRC 6-3, 6-38, 6-46, 14-3
Secondary Flow-Through Control 6-2, 6-26, 7-3
Serial EEPROM 5-7, 6-29–6-30
SFTCR 6-26, 7-3, 18-1, 18-2
SINCPCNT 6-23, 6-24, 7-3, 18-1, 18-3, 19-2, 19-4
SITLPCNT 6-23, 6-24, 7-3, 18-1, 18-2, 19-2, 19-4
SMAXPCNT 6-24, 6-25, 6-27, 7-3, 18-1, 18-3, 19-3
SPUBARD*x* registers 6-43–6-45
SPUL32BAR1 6-41
SPUL32BAR2 6-42
SPUL32BAR3 6-43
SPUU32BAR1 6-42
SPUU32BAR2 6-42
SPUU32BAR3 6-43
System Error Event 6-32
TEST 6-29
Timeout Control 6-2, 6-19, 7-3, 8-4
Timer 4-6, 6-15, 6-31
TMRCNT 4-6, 6-31
TMRCNTRL 4-6, 6-31
TOCNTRL 6-19, 6-32, 7-3, 8-4
VPD 2-1, 6-49, 21-1
reset
5-1–5-7
JTAG 22-1, 22-2
pins 3-3, 3-18
reset input effect 5-4
resistor recommendations, pull-up/pull-down 3-2–3-4
rotating-priority scheme
14-2

RRC register
6-3, 6-38, 6-46, 14-3
RSTIN# 3-20, 15-1
rules
deadlock avoidance 11-5
PCI-X Bus, general 9-1–9-4
transaction ordering, conventional PCI 11-1–11-3
transaction ordering, PCI-X 11-5

S

S_ACK64# 3-2, 3-10, 5-5
S_AD[31:0] 3-10, 16-4
S_AD[63:0] 3-2, 5-5
S_AD[63:32] 3-10, 6-38
S_AV_{DD} 3-24, 4-4, 5-5, 23-3
S_AV_{SS} 3-24, 5-5
S_CBE[3:0]# 3-11, 16-3–16-4
S_CBE[7:0]# 3-2, 5-5
S_CBE[7:4]# 3-11, 6-38
S_CFN# 3-22, 5-5, 6-29, 14-1
S_CLKIN 3-3, 3-16, 4-1, 4-3, 4-4, 4-6, 5-5, 6-29, 6-31, A-2
S_CLKIN_STB 3-3, 3-16, 4-4, 5-5
S_CLKO[4:0] 4-1–4-4, 5-5, 6-48
S_CLKO[4:1] 3-3, 3-16, 3-17, 4-1
S_CLKO0 3-3, 3-16, 3-17, 6-34, B-2
S_CLKOFF 3-3, 3-17, 4-1, 5-5
S_CLKRUN# 3-22, 5-5, 6-36
S_CR 3-3, 3-17, 5-5
S_DEVSEL# 3-2, 3-11, 5-3, 5-5, 12-1, 17-1, 17-2
S_FRAME# 3-2, 3-11, 5-5, 13-2, 14-2, 14-3
S_GNT[7:0]# 3-2, 5-6, 14-1
S_GNT[7:1]# 3-12, 14-1
S_GNT0# 3-11, 14-1, 14-3
S_IDSEL 10-5
S_IRDY# 3-2, 3-12, 5-6, 6-18, 14-3
S_LOCK# 3-2, 3-12, 5-6, 13-1–13-2
S_M66EN 3-2, 3-12, 4-4, 5-1, 5-2, 5-6
S_PAR 3-2, 3-12, 5-6, 14-3
S_PAR64 3-2, 3-13, 5-6, 6-38, 14-3, 17-2
S_PERR# 3-2, 3-13, 5-6, 6-10, 12-1–12-13
S_PLLEN# 3-3, 3-17, 4-4, 5-6, 23-3
S_REQ[7:0]# 3-2, 5-6, 6-17, 14-1
S_REQ[7:1]# 3-13, 14-1
S_REQ0# 3-13, 14-1, 14-3
S_REQ64# 3-2, 3-14, 5-6
S_RSTOUT# 3-18, 4-3, 4-4, 5-2, 5-3, 5-4, 5-6, 6-15, 6-17, 6-51, 20-1
S_SERR# 3-2, 3-14, 6-14, 12-1, 12-5, 12-12–12-13
S_STOP# 3-2, 3-14, 5-3, 5-6
S_TRDY# 3-2, 3-14, 5-3, 5-6
S_TST[1:0] 3-22, 5-6
S_V_{IO} 3-4, 3-24, 5-6
S_XCAP_IN 3-23, 5-1, 5-6, 9-1
S_XCAP_PU 3-23, 5-1, 5-6
SAC 8-2, 9-2, 9-11, 10-3
Secondary
clock frequency measurement 4-6
Clock pins 3-3, 3-16, 3-17, 3-21, 3-22, 4-1, 4-4, 4-6, 5-2, 20-1
Flow-Through Control register 6-2, 6-26, 7-3
PCI Bus Interface pins 3-2, 3-10–3-14
Serial EEPROM 6-2, 7-1–7-4
pins 3-3, 3-19
registers 5-7, 6-29–6-30
SFTCR register 6-26, 7-3, 18-1, 18-2
signal specs 24-1–24-4
signaling voltage 3-24
SINCPCNT register 6-23, 6-24, 7-3, 18-1, 18-3, 19-2, 19-4
Single Address Cycle
See SAC
SITLPCNT register 6-23, 6-24, 7-3, 18-1, 18-2, 19-2, 19-4
smart prefetch 6-27, 6-40–6-45, 19-3
SMAXPCNT register 6-24, 6-25, 6-27, 7-3, 18-1, 18-3, 19-3
specs
electrical 23-1
mechanical 24-1–24-4
split completion 9-1–9-20
SPUBARD*x* registers 6-43–6-45
SPUL32BAR1 register 6-41
SPUL32BAR2 register 6-42
SPUL32BAR3 register 6-43
SPUU32BAR1 register 6-42
SPUU32BAR2 register 6-42
SPUU32BAR3 register 6-43
System Error Event register 6-32

T

TAP controller
See test access port controller
target abort
See abort, target
TCK 3-19, 22-1
TDI 3-19, 22-1, 22-2
TDO 3-19, 22-1, 22-2
termination, abnormal 13-2
test access port controller
22-1, 22-2

TEST register
6-29
testability 22-1–22-2
timeout
19-3
control 6-2, 7-3
Timeout Control register 6-2, 6-19, 7-3, 8-4
Timer registers 4-6, 6-15, 6-31
TMRCNT register 4-6, 6-31
TMRCNTRL register 4-6, 6-31
TMS 3-19, 22-1
TOCNTRL register 6-19, 6-32, 7-3, 8-4
transactions
PCI 8-1–8-18, 11-1–11-3
PCI-X 9-1–9-22, 11-4–11-6
TRST# 3-19, 22-1, 22-2

V

V_{DD_CORE} 3-24, 5-6
V_{DD_IO} 3-24, 5-6
VGA 6-14, 10-1, 10-4, 10-4–10-5
VHDL 22-2
VHSIC Hardware Description Language 22-2
voltage signaling 3-24
VPD 21-1
registers 2-1, 6-49, 21-1
V_{SS}
3-15, 3-17, 3-24, 5-6