AMCC

APPLIED MICRO CIRCUITS CORPORATION

PCI PRODUCTS
DATA BOOK

Please refer to AMCC's website at www.amcc.com for the latest Device Summary information for the S5920 and S5933 PCI products.

For Marketing and Application Information Contact:

Applied Micro Circuits Corporation
6290 Sequence Drive
San Diego, CA 92121-4358
(800) 755-2622
(619) 450-9333
Fax (619) 450-9885
http://www.amcc.com

The material in this document supersedes all previous documentation issued for any of the products included herein.

AMCC reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

AMCC does not assume any liability arising out of the application or use of any product or circuit described herein, neither does it convey any license under its patent rights nor the rights of others.

AMCC reserves the right to ship devices of higher grade in place of those of lower grade.

AMCC SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

AMCC and Matchmaker are registereded trademarks of Applied Micro Circuits Corporation.

All other product names are trademarks, registered trademarks, or servicemarks of their respective owners.

PRINTED IN U.S.A./PCIPROD-0498

CONTENTS

SECTION 1: General Information

VISION

It is AMCC's vision to be the premier supplier of silicon for high bandwidth connectivity solutions for the worldwide networking infrastructure.

CORPORATE OVERVIEW

AMCC defines, develops, manufactures and markets application specific standard products (ASSPs) and customer specific integrated circuits (CSICs) for high speed, high performance network interface applications. Utilizing CMOS, BiCMOS and proprietary MicroPower Bipolar technology, AMCC provides precision circuits and interface solutions for Gigabit Ethernet, ATM, SONET, Fibre Channel and PCI markets.

Since 1979, AMCC has designed and produced five generations of semicustom bipolar ECL logic arrays and two generations of BiCMOS logic arrays. AMCC expertise includes its mixed ECL/TTL interface, phase-locked loop (PLL), precision vernier, skew control, high-speed VCO and controlled edge rate TTL outputs.

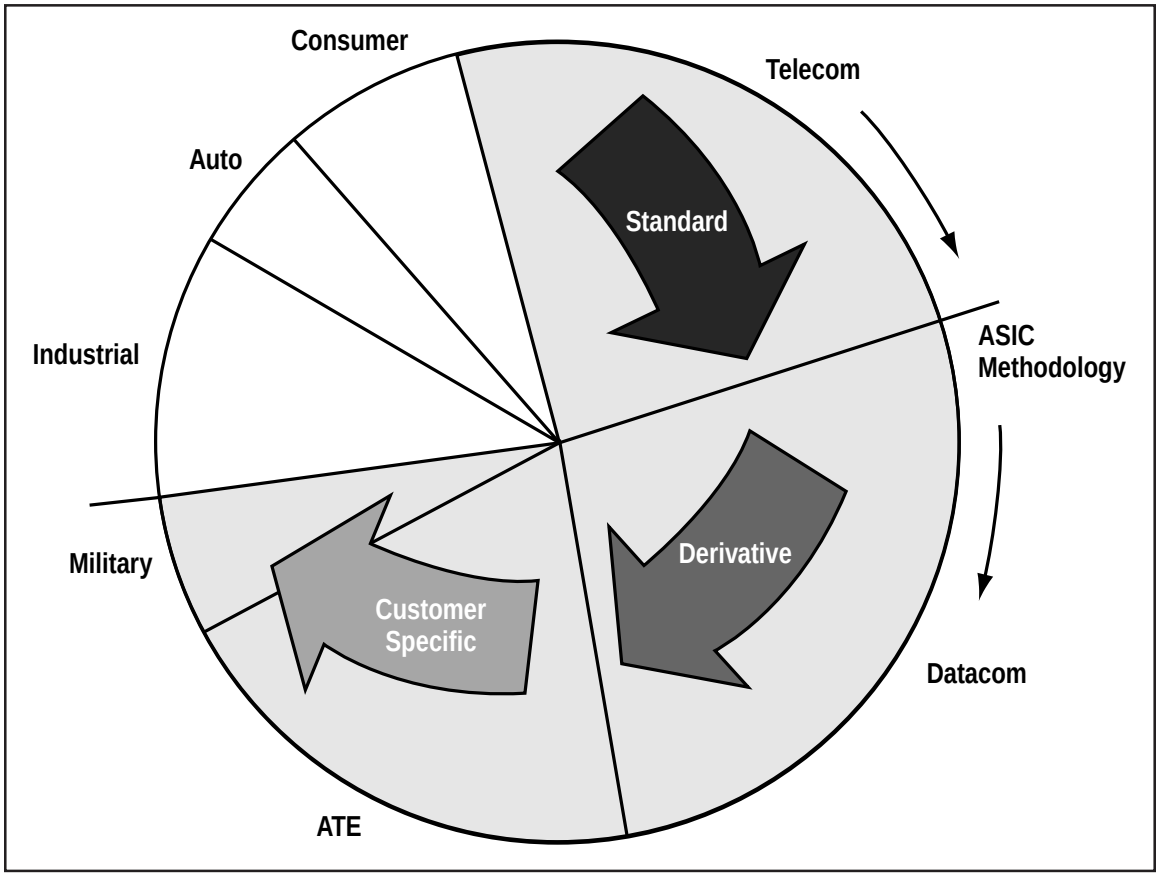
AMCC Product Development Strategy

AMCC's product development strategy utilizes the company's expertise in the telecommunications, data communications, ATE, computer and military markets. Initially ASSPs are defined based on key industry standards. Then, as additional applications are identified, derivative products based on the "cores" of the original devices are introduced. These "cores" are then made available for high volume proprietary customer specific designs. This is all made possible by our emphasis on using ASIC techniques and methodologies to develop the original products.

Network Interface Products

High performance network interface encompasses a wide range of applications, all requiring data transmission rates from >100 Mbit per second to over 9.6 Gbit per second. These applications include Local Area Networks (LAN), Hubs, Routers and Switches, Wide Area Networks (WAN) based on SONET standards, RAID-based storage systems, serial backplane for high performance switches and digital video broadcasting systems.

AMCC Product Development Strategy



CAPABILITY SUMMARY

AMCC interface circuits, transceiver chips and switches are designed to implement emerging network technologies such as the ANSI Fibre Channel and High Performance Interface (HIPPI) standards, the ITU SONET telecommunications standard, the ATM Forum LAN standard and IEEE Gigabit Ethernet standard. Jitter, speed, power and size are critical design issues for all these circuits. AMCC's devices are based on its unique Bipolar process which has noise isolation characteristics that enable best-in-class jitter performances. The inherent physical structure of the company's process makes 1 to 3 GigaHertz (GHz) data rates possible at low power. Consequently, the low device power consumption of AMCC's products helps to minimize the cost and size of packaging.

Peripheral Component Interconnect (PCI) Bus Controllers

Increasing bandwidth of high speed networks creates a bottleneck at the desktop. One of the causes of this problem is the latency associated with connection to high speed peripheral equipment over LANs and WANs. The 132 Megabyte per second backplane PCI bus helps break the bottleneck.

AMCC has developed the industry's first line of general purpose master/slave controllers for the PCI bus. These circuits provide a high performance single-chip interface for add-on boards and adapter cards for industrial, graphics, video and communications markets.

Precision Clock and Timing Products

AMCC provides a line of high precision clock and timing standard products for exacting system designs. AMCC has also tailored clock and timing devices to specific customer needs for high performance clock generation and distribution, clock synchronization and de-skewing, frequency synthesis, and pulse shaping applications. Offerings include low EMI, low skew clock drivers and low jitter clock generators for high performance server, workstation and RAMBUS™-based applications.

Manufacturing Excellence

AMCC manufactures its own Bipolar and BiCMOS wafers using proven processes and utilizes foundry relationships for access to CMOS capability in its world class fab located in San Diego. AMCC is proud of its best-in-class status for lowest defect densities for like sized fabs. This allows AMCC to provide a continuous, predictable supply of product to its customers.

The Company follows a "semi-fabbed" manufacturing strategy and has CMOS and BiCMOS foundry relationships in place with major domestic and international semiconductor partners that provide for significant additional production capacity. Wafer purchases from strategic foundry partners both expand capacity and provide alternate sources. Additional high-volume assembly and test facilities are located offshore.

AMCC's "quick turn, semi-fabbed" manufacturing approach blends together the strengths of both the "fabbed" and "fabless" semiconductor strategies. Fabbed advantages include the security of total in-house control and time to market. Fabless advantages include multiple sourcing and allows the company to focus investment on new high performance products.

AMCC COMMITMENT TO QUALITY

AMCC is committed to achieving the highest quality and reliability level in the integrated circuit products we provide. Every year for over a decade we have established industry-leading reliability and outgoing quality targets and then exceeded them.

The quality and reliability philosophy at AMCC starts with the premise that for AMCC to continue to excel and be the premier supplier to our customers, the quality expectations of customers must be consistently met or exceeded.

Our team operating philosophy is to:

1) design in manufacturability and reliability during the new product development phase (plan);

2) build in quality at all manufacturing steps (do);

3) execute thorough product inspections, internal audits and reliability confirmation (check);

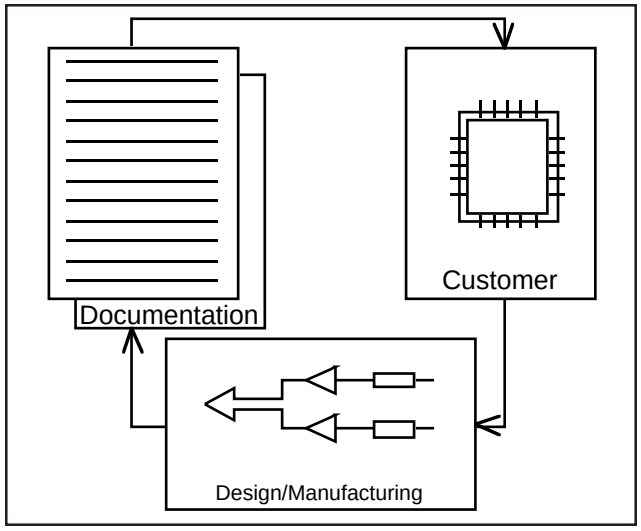
4) incorporate feedback from internal and external sources into continuous quality improvement programs (act).

Reliability and Manufacturability—Designed In From The Start

Reliability and manufacturability is designed in up front through a team infrastructure which focuses on active participation by Design, Manufacturing and Reliability Engineering throughout all phases of the design process. This includes extensive design verification through computer modeling and design validation by product characterization and application simulation. Final team design review and production readiness approval is required prior to release of products to production.

Quality Built In During Wafer Fabrication and Manufacturing

AMCC's manufacturing and quality teams employ documented operating procedures, work instructions, in-process inspections and SPC methodology to provide assurance of continued process control and compliance to specification.



QA gates and subsequent feedback ensures quality confirmation of AMCC's final product in a continuous improvement program.

Inspection, Audit and Reliability Confirmation

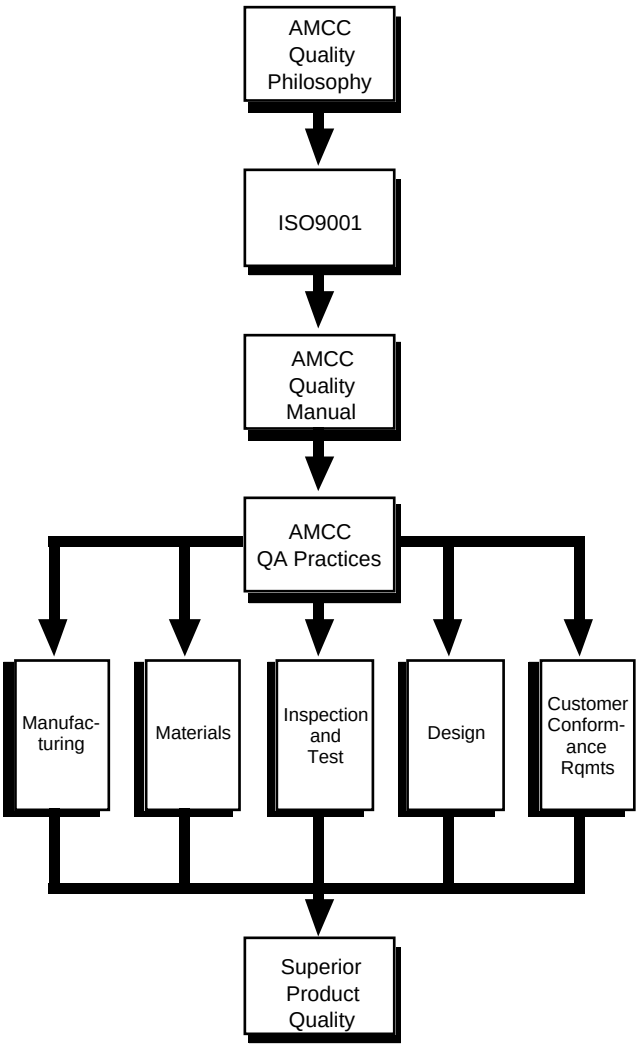
AMCC has strategically placed In-Process Quality Control (IPQC) gates, internal process/area audits and lot/time specific reliability monitors to verify performance against customer requirements and internal design/manufacturing process capabilities. Metrics generated by these activities are intended to provide continuous improvement feedback data for review and action as driven by senior management.

- Die visual and precap gate
- Final outgoing inspection gate
- Modified MIL-STD-105D sampling program
- Lot specific group A and B testing
- Ongoing reliability monitors
- SPC/Data metric review of key subcontractors
- Visual/mechanical and electrical outgoing indices and PPM goals
- Cost-of-quality pareto analysis

QUALITY SYSTEM OVERVIEW

Continuous Quality Improvement Program

- Corporate-wide commitment driven by the Executive Staff
- A program plan that is flexible enough to comprehend dynamic customer inputs
- Statistical tools in place for analysis and action planning
- Weekly and monthly review meetings to share performance data
- Self examination consistent with elements in ISO9001 and the *Malcolm Baldridge National Quality Award*



AMCC QUALITY SYSTEM

The Quality System had been modeled after the stringent military requirements of MIL-I-45208, MIL-Q-9858 and MIL-I-38535 Appendix A. Heading into the 21st century, AMCC has now modified its Quality System to also align with ISO9001. This has strengthened the closed loop improvement cycle by tying internal audits with corrective/preventative action though continuous management review.

AMCC's Quality System has the following components integrated throughout the factory to meet or exceed the above requirements.

- Quality Organization
- Quality Planning
- Management Review
- Contract Review
- Design Control
- Document and Data Control
- Purchasing
- Supplier Selection and Control
- Control of Customer Supplied Materials
- Product Identification and Traceability
- Operating Procedures
- Work Instructions
- Inspection and Test
- Inspection, Measurement and Test Equipment Calibration
- Inspection Status System
- Control of Nonconforming Material
- Corrective and Preventive Action
- ESD Safe Handling, Storage, Packaging, Preservation and Delivery Methods
- Records Retention and Maintenance
- Internal Process/Area Auditing System
- Training/Certification
- SPC and Statistical Techniques
- Failure Analysis

ISO9001 REGISTRATION

Based on the restructure of the Quality System to ISO9001 requirements and successful completion of internal and third-party audits, AMCC was ISO registered on July 29, 1996. Bi-yearly surveillance audits have been successfully passed as well. Please contact the factory for further details and schedule updates.

PRODUCT QUALIFICATIONS

A qualification is a sequence of tests in which all parameters, including the reliability of the device are tested. It is this sequence of tests which **initially qualifies** the part to be released for production.

Thorough reliability testing is performed on new product and package families in order to ensure the expectations of our customers are met. These tests include environmental, mechanical and life testing performed in accordance with Military Standards, industrial accepted methods and AMCC Test Procedures. Contact the factory for specific details regarding your selected product/package combination.

AMCC provides MIL-STD-883 Methods 5005 and 5010 testing for our military customers on contract as well as MIL-H-38534 quality conformance screening for hybrid customers.

MIL-STD-883 Method 5005

"Qualification And Quality Conformance Procedures"

Method 5005 establishes qualification and quality-conformance inspection procedures for semi- conductors to ensure that the quality of devices and lot conform with the requirements of the applicable procurement document. The full requirements of Group A, B, C, D, and E test and inspections are intended for use in initial device qualification—or requalification in the event of product or process change—and in periodic testing for retaining qualification.

Group A consists of electrical tests performed on an inspection lot which has already passed the 100% screening requirements. After a lot has passed the 100% screen tests, a random sample of parts is selected from the total population of devices to form the inspection lot. The inspection lot is then subjected to these Group A electrical tests.

Group B inspection tests are used to monitor the fabrication and assembly processes performed on each inspection lot.

Group C consists of a 1000-hour life test conducted to verify die integrity.

Group D verifies the material integrity and the reliability of the package.

Group E demonstrates the radiation hardness capability of the device. Performed on a generic basis by device type or as required for an application.

MIL-STD-883 Method 5010

"Test Procedures For Custom Monolithic Microcircuits"

This method establishes screening and quality conformance procedures for the testing of custom and semicustom monolithic semiconductors to verify Class B or Class S quality and reliability levels. Testing is performed in conjunction with other documentation such as MIL-I-38535 and an applicable detail specification. It establishes the design, material, performance, control, and documentation requirements needed to achieve prescribed levels of device quality and reliability. AMCC can support qualification using this method.

Until August of 1983, the qualification most commonly used was Method 5005. Since that time, the newer revision of MIL-STD-883 includes Method 5010, which is better suited for semicustom devices (logic arrays included). Either qualification is adequate, but it is desirable to use the 5010 qualification procedure in qualifying custom or semicustom devices.

Qualification Method 5005 VS. 5010

The primary difference between the two methods is in the Group D test. Method 5005 uses electrically-good devices, where method 5010 uses electrical rejects and package-only parts for environmental tests. In addition, Method 5010 is designed for smaller production releases (i.e., 2000 devices/year) while Method 5005 is designed for large production releases.

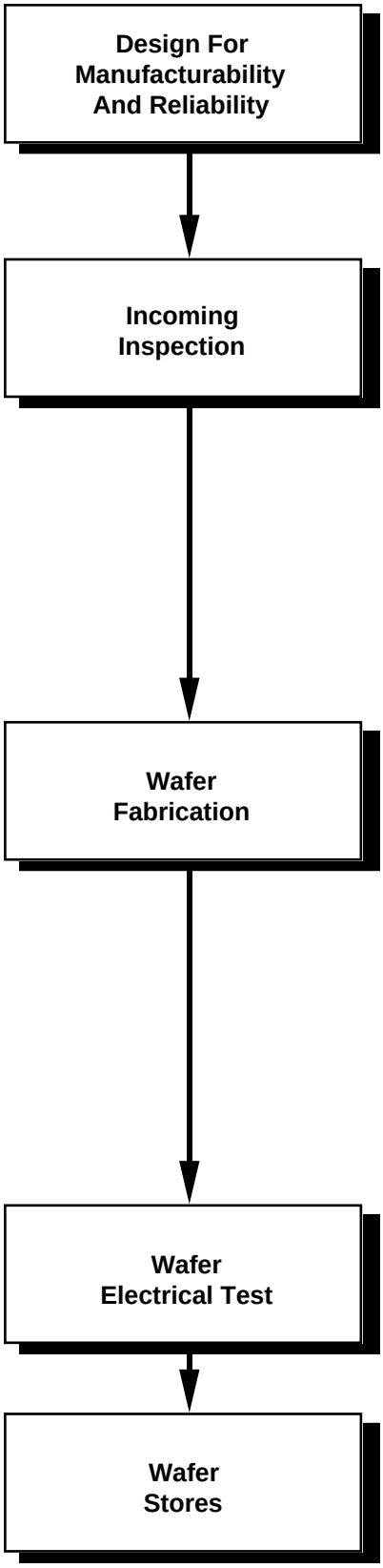
Generic Data

Under the provision of MIL-I-38535, a customer can elect to qualify using generic data (similar device/family). However, the provisions of the applicable contract should be reviewed. In most cases generic data will satisfy full qualification requirements.

Since many of the qualifications at AMCC are ongoing, generic data may be available for this purpose.

QUALITY SYSTEM OVERVIEW

AMCC Product Assurance
Product Flow Detail

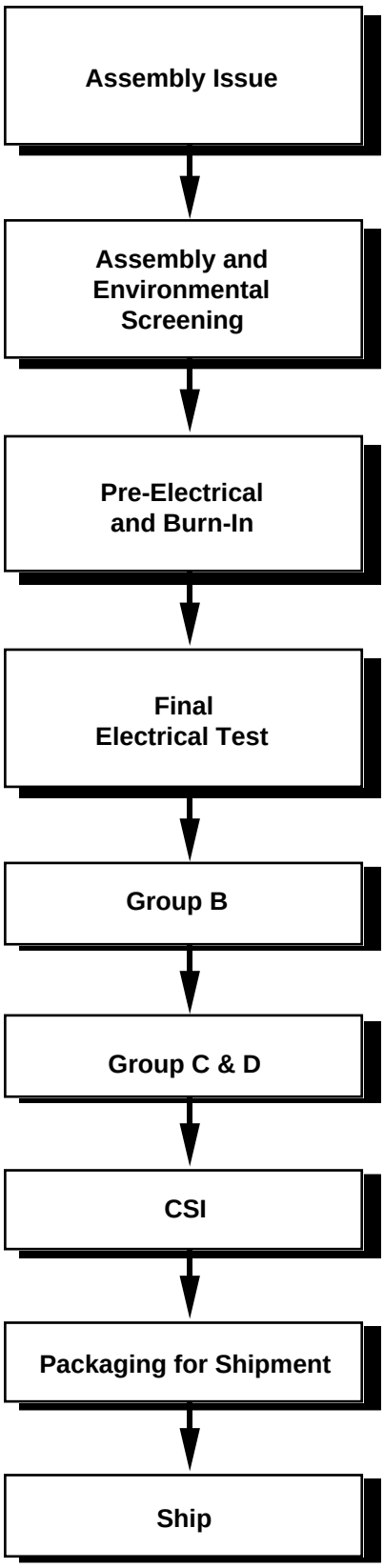


- Component Selection
  - Definition — Specification
  - Supplier: Selection — Qualification — Approval
  - Qualified Vendors List (QVL)

- Acceptance Documents and Operating Procedures
  - Purchase Order
  - Component/Material Specifications
  - Product Assurance and General Procedures
  - QVL
  - Sample Inspection of All Direct Materials

- Class 10 Clean Room — FED-STD-209
- Measurements in Adherence with MIL-STD-977
- SPC:
  - In-process Monitors
  - PCM Electricals
- SEM Inspection on All Military Lots
- QA Audits
  - CV Plots — Weekly (minimum)
  - DI Water — Weekly
    - Particle Counts
    - Bacteria Count
  - Airborne Particle Count — Weekly

AMCC Product Assurance Product Flow Detail



- IPQC Audit
  - Manufacturing Procedures

Hermetic Military Flow	Hermetic Commercial Flow	Plastic Flow

• Method 5004[1]
• IPQC
  • Customer Source Inspection (CSI)
  • Assembly Final Inspection
• Methods 5004 & 5005[1]
	• Method 5004 (No Centrifuge
	• IPQC
	  • Assembly Final Inspection
		• Precap Visual
		  • Die Shear SPC
		  • Wire Bond SPC
		• Mold/External Visual
		• Lead Trim and Co-planatary Inspection

• Methods 5004 & 5005[1] • IPQC • Group A	• 25°C — 100% • 0°C & 70°C — AQL=0.25% Sample	

• Method 5005[1]
• IPQC
  • Final Inspection
• Method 5005[1]
• Performed by Customer Order
	• IPQC
	  • Final Inspection
	  • Coplanatary Check	

• Per Order		

• IPQC
  • Outgoing Inspection
	• IPQC
	  • Outgoing Inspection	

[1]MIL-H-38534 or MIL-STD-883 method 5010 also available.

AMCC'S RELIABILITY VIGIL

AMCC's internal reliability vigil consists of three phases:

- New/changed processes and material qualifications
- In-process Quality monitors
- Periodic operating life and environmental testing

New/Changed Wafer Processes and Material Qualifications

In order to initially release a device to production a standard set of MIL-STD-883 tests must be completed successfully. These tests include:

Wafer Process and Design

- Operating Life Method 1005
- ESD Characterization Method 3015
- Wire Bond Pull Method 2011
- Thermal Shock or Method 1011 or 1010 Temperature Cycling

Package and Related Materials

- Selected Subgroups of MIL-STD-883, Method 5005, Group B and D

AMCC adheres to MIL-I-38535 with regards to changes.

If changes to production released devices are determined to be major, the appropriate qualification testing must be successfully completed prior to change approval.

In-Process Quality Monitors

- CV plots
- Airborne particle count
- Bacteria, particle count, and resistivity on DI water
- ESD work stations and procedures
- In-line testing of process gases
- Temperature and humidity control
- SPC in wafer fabrication
- SEM of all military lots

Periodic Operating Life and Environmental Testing

- Performed on a product from each process family quarterly.
- 1000 hour operating life test (minimum), Method 5005, Group C.
- Temperature cycling per Method 1010, 100 cycles, condition C: –65°C/150°C
- Environmental testing per AMCC standard test procedures. Consult factory for further details.

Final Measure and Assurance of Quality

The cost of defects depends on when the failure occurs. For example, costs rise significantly as undetected defective ICs are integrated into systems. High quality parts cut costs substantially, and the extra quality built into every AMCC device means added value to our customers.

To achieve maximum quality, AMCC employs 100% testing of all devices, followed by stringent QA sampling.

AMCC performs QA sampling measurements at full specification temperature, both DC and AC, to achieve the tightest AQLs in the industry.

RADIATION HARDNESS

High energy radiation can cause structural changes in the silicon and silicon dioxide crystal lattice by displacing atoms from their normal crystal sites. These changes can be responsible for increased junction leakage, degraded transistor current gain (b), and increased parasitic Si/SiO2 interface leakage currents. The damage is generally induced by neutrons, X-rays, and gamma rays. The effects of the damage induced by this radiation can change both AC and DC parameters, affect functional performance, and, in severe cases, destroy the device.

Certain of AMCC's high performance products are inherently radiation resistant. The radiation resistance of AMCC IC's is the result of the small geometries, the structure of the fabrication process itself, and the use of ECL logic within the device. Contact your AMCC representative regarding radiation resistance characteristics associated with a specific product.

ATM LAN and 100VG AnyLAN Products

Products	Function	Operating Speed	Data Path	Package	Power Supply
S3011	SONET/ATM/ E-4 Tx	139/155 Mbit/s	8:1 bit	80 TEP	+5V
S3012	SONET/ATM/ E-4 Rx	139/155 Mbit/s	1:8 bit	80 TEP	+5V
S3020	ATM Tx	622 Mbit/s	8:1 bit	52 TEP	+5V
S3021	ATM Rx	622 Mbit/s	1:8 bit	52 TEP	+5V
S3027	Clock Recovery	155/622 Mbit/s	1 bit	20 TSSOP	+5V
S3028	ATM Transceiver	155/622 Mbit/s	1:8/8:1 bit	64 PQFP	+5V
S3029	Quad Transceiver	155 Mbit/s	1 bit	64 TQFP	+3.3V
S2100	100VG AnyLAN Transceiver	120 Mbit/s	4:1/1:4 bit	52 PQFP	+5.0V

PRODUCT SELECTION GUIDES

Fibre Channel/Gigabit Ethernet Products

Products	Function	Operating Speed	Data Path	Package	Power Supply
S2036	Open Fiber Control	266/531/1062 Mbit/s	N/A	28 SOIC	+5V
S2042	10-bit Compliant Fibre Channel	266/531/1062 Mbit/s	10:1/20:1 bit	52 PQFP	+5V/+3.3V
S2043	10-bit Compliant Fibre Channel	266/531/1062 Mbit/s	1:10/1:20 bit	52 PQFP	+3.3V
S2044	GLM Compliant Fibre Channel Tranmitter	266/531/1062 Mbit/s	10:1/20:1 bit	52 PQFP	+3.3V
S2045	GLM Compliant Fibre Channel Receiver	266/531/1062 Mbit/s	1:10/1:20 bit	52 PQFP	+3.3V
S2046	Gigabit Ethernet Transmitter	1250 Mbit/s	10:1/20:1 bit	52 PQFP	+3.3V
S2047	Gigabit Ethernet Receiver	1250 Mbit/s	1:10/1:20 bit	52 PQFP	+3.3V
S2052	10-bit Compliant Fibre Channel	1062/1250 Mbit/s	10:1/1:10 bit	64 PQFP	+3.3V
S2053	Gigabit Ethernet and Fibre Channel Transceiver	1062/1250 Mbit/s	10:1/1:10 bit	64 PQFP	+3.3V
S2054	Gigabit Ethernet and Fibre Channel Transceiver	1062/1250 Mbit/s	10:1/1:10 bit	64 PQFP	+3.3V
S2057	Port Bypass Circuit	Up to 1500 Mbit/s	1:1 bit	20 TSSOP	+3.3V
S2058	Port Bypass and Repeater	1062 Mbit/s	1:1 bit	28 SOIC	+3.3V

See Network Products data book or http://www.amcc.com.

HIPPI Products

Products	Function	Operating Speed	Data Path	Package	Power Supply
S2020	HIPPI Source	800 Mbit/s	32 bit	225 PGA/ 208 TEP	-5.2/+5V
S2021	HIPPI Destination	800 Mbit/s	32 bit	225 PGA/ 208 TEP	-5.2/+5V

See Network Products data book or http://www.amcc.com.

PCI Products

Products	Function	Description
S5920	Target PCI Interface	General Purpose PCI Bus Interface Chip
S5920DK1	Developer's Kit	Software/Hardware ISA Bus, SRAM,PLD and User Developer Kit
S5933	Master/Slave PCI Controller	General Purpose PCI Bus Controller
S5933DK1	Developer's Kit	Software/Hardware PLD and User Prototype Developer Kit
S59CD-ROM1	Software CD-ROM	S5920/S5933 Developer Kit Software Code Examples, Drivers and Utility Programs

SONET/SDH/ATM Products

Products	Function	Operating Speed	Data Path	Package	Power Supply
S3005	SONET/ATM/ E-4 Tx	139/155/622 Mbit/s	8:1 bit	68 LDCC 80 TEP	-4.5/5.0V
S3006	SONET/E-4 Rx	139/155/622 Mbit/s	1:8 bit	68 LDCC 80 TEP	-4.5/5.0V
S3014	Clock Recovery	155/622 Mbit/s	1 bit	44 PLCC	-5.2/+5.0V
S3015	E4/OC-3/STM-1 Tx	139/155 Mbit/s	1 bit	52 TEP	+5V
S3016	E4/OC-3/STM-1 Rx	139/155 Mbit/s	1 bit	52 TEP	+5V
S3017	SONET/SDH Tx	622 Mbit/s	8:1 bit	52 TEP	+5V
S3018	SONET/SDH Rx	622 Mbit/s	1:8 bit	52 TEP	+5V
S3019	SONET/SDH XCVR w/CDR	155/622 Mbit/s	8:1/1:8 bit	80 PQFP	+3.3V
S3025	Clock Recovery	622 Mbit/s	1 bit	20 TSSOP	+5V
S3026/27	Clock Recovery	155/622 Mbit/s	1 bit	20 TSSOP	+5V
S3028	SONET/SDH Transceiver	155/622 Mbit/s	8:1/1:8 bit	64 PQFP	+5V
S3029	Quad Transceiver	155 Mbit/s	1 bit	64 TQFP	+3.3V
S3030/31	E4/OC-3/STM-1 ATM XCVR	139/155 Mbit/s	1 bit; 4:1/1:4 bit	100 PQFP	+5V
S3032	SONET/SDH/ATM XCVR w/CDR	155/622 Mbit/s	8:1/1:8 bit	64 PQFP	+3.3V
S3033	SONET/SDH/ATM XCVR	155/622 Mbit/s	8:1/1:8 bit	64 TQFP	+3.3V
S3035	SONET/SDH/ATM XCVR w/Dual I/O	155/622 Mbit/s	8:1/1:8 bit	80 PQFP	+3.3V

See Network Products data book or http://www.amcc.com.

SONET/SDH/ATM Products (continued)

Products	Function	Operating Speed	Data Path	Package	Power Supply
S3040	SONET/SDH Clock Recovery Unit	2.5 Gbit/s	1 bit	32 TQFP	+5.0V
S3041	SONET/SDH Transmitter	2.5 Gbit/s	8:1 bit	100 TQFP	+3.3V
S3042	SONET/SDH Demux	2.5 Gbit/s	1:8 bit	100 TQFP	+3.3V
S3043	SONET/SDH Transmitter	2.5 Gbit/s	16:1 bit	80 PQFP	+3.3V
S3044	SONET/SDH Demux	2.5 Gbit/s	1:16 bit	80 PQFP	+3.3V
S3045	SONET/SDH OC-12 to OC-48	77/311 Mbit/s	32:8/8:32bit	208 PQFP	+3.3V
S3047	SONET/SDH Clock and Data Recovery w/no Ref. Clock	2.5 Gbit/s	1 bit	32 TQFP	+5.0V

See Network Products data book or http://www.amcc.com.

Crosspoint Switch Products

Products	Function	Operating Speed	Data Path	Package	Power Supply
S2016	Crosspoint Switch	1.5 Gbit/s	16 X16	120 TEP	+5V
S2024	Crosspoint Switch	600/800 Mbit/s	32 X 32	196 LDCC	-5.2/+5V
S2025	Crosspoint Switch	1.5 Gbit/s	32 X 32	196 LDCC	+5V
S2028	Crosspoint Switch	1.5 Gbit/s	33 X 32	224 LDCC	+3.3V

See Network Products data book or http://www.amcc.com.

PRODUCT SELECTION GUIDES

Precision Clocking Products

	Output Frequency with Respect to Input Frequency					
P/N	**Total Outputs**	**Number of Outputs Divide-by-1**	**Number of Outputs Divide-by-2**	**Power Supply**	**Output Levels**	**Package**
SC3506	20	10	10	+5V	TTL	52 PQFP
SC3306	20	10	10	+5V	LVTTL	52 PQFP
SC3308	20	20	N/A	+5V	LVTTL	52 PQFP
SC3318	10	10	N/A	+5V	LVTTL	28 SOIC
SC3368	14	6	8	+5V	LVTTL	28 SOIC
S3LV308	20	20	N/A	+3.3V	LVTTL	52 PQFP

See Network Products data book or http://www.amcc.com.

Clock Generator and Synthesizer Products

	Output Frequency with Respect to Input Frequency						
P/N	**Description**	**Input Reference**	**Number**	**Type**	**Maximum Frequency**	**Minimum Delay Adjust Increment**	**Number of Selectable Output Relationships**
S4402	Multiphase Clock Generator	TTL	6	TTL	80	3.2ns	21
S4403	Multiphase Clock Generator	TTL	10	TTL	80	3.2ns	21
S4405	Multiphase Clock Generator	PECL/TTL	6 1	TTL PECL	80 160	3.2ns -	21 -
S4406	Clock Generator with delay Adj.& Invert	TTL	12	TTL	66	4 ns @ 66 MHz	7x4 Banks of 3 Outputs
S4503	Clock Synthesizer	XTAL	2 1	TTL PECL	80 300	N/A	Multiply 2-32 Divide 2-16
S4506 S4507	RAMBUS-TM Compatible Clock Generator	XTAL	2	Rambus Compatible	250 300	N/A	1

See Network Products data book or http://www.amcc.com.

PRODUCT SELECTION GUIDES

ASIC Standard Cell Products

Family Name	Technology	Operating Speed	Equivalent Gates	Number of I/O	Analog Functions	Power Supplies Options
Micro-power	1 Micron Bipolar	Up to 2.5GHz	Up to 4000	Up to 200	• 2.5 GHz PLL • 1 GHz Timing Vernier • Custom Analog	• +5V • +5V/-5V • -5V • +3.3V

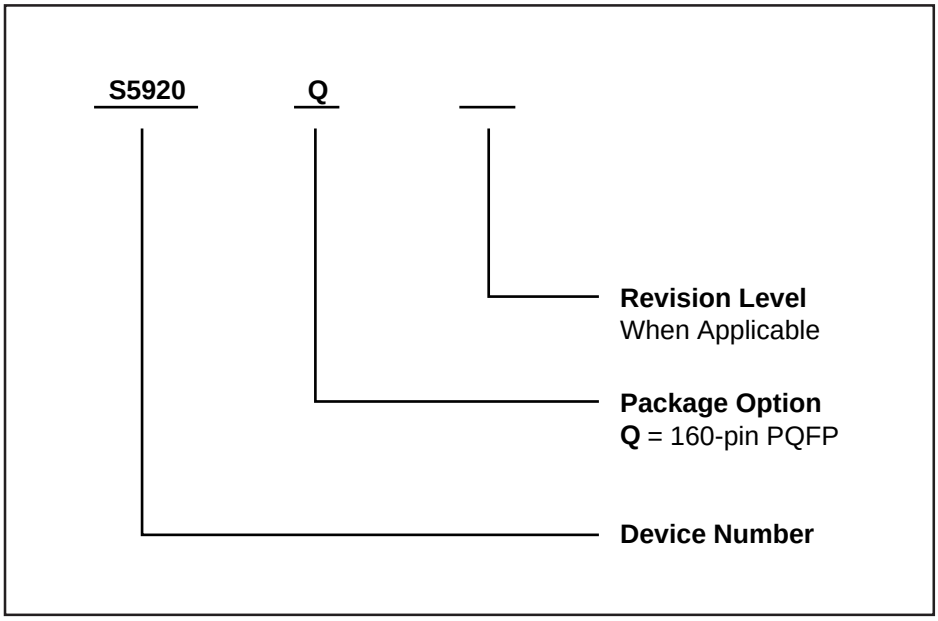
See Network Products data book or http://www.amcc.com.

ASIC Logic Array Products

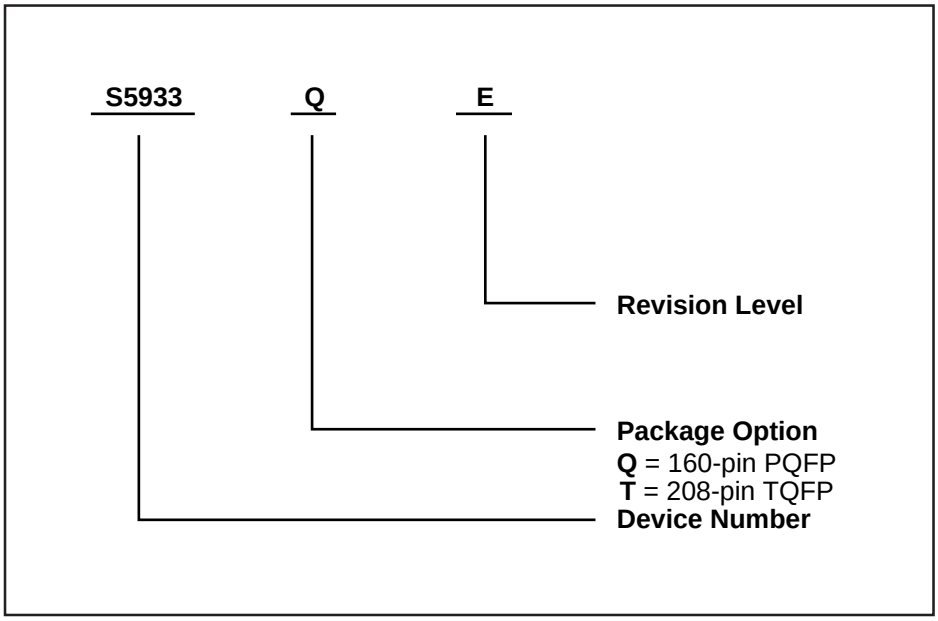
Part Number	Technology	Equivalent Gates (Full Adder Method)	Number of I/O	Stuctured Arrayed Blocks
Q20004	1 Micron Bipolar	671	30	None
Q20010	1 Micron Bipolar	1469	68	None
Q20025	1 Micron Bipolar	4032	102	None
Q20045	1 Micron Bipolar	6782	130	None
Q20080	1 Micron Bipolar	11242	164	None
Q20120	1 Micron Bipolar	18777	200	None
Q20P010	1 Micron Bipolar	973	34	1.25 GHz PLL
Q20P025	1 Micron Bipolar	3272	51	1.25 GHz PLL
Q20M100	1 Micron Bipolar	13475	195	RAM

See Network Products data book or http://www.amcc.com.

S5920 Ordering Information



S5933 Ordering Information



SECTION 2: S5920 PCI Target Interface

2

CONTENTS

FEATURES

- Full 132 Mbytes/sec Transfer Rate
- PCI Bus Operation to 33 MHz
- PCI Purposed 2.2 Compliant Target/Slave Device
- Add-On Bus up to 40 MHz
- Programmable Prefetch and Wait States
- 8/16/32-Bit Add-On Bus
- Four Definable Pass-Thru Regions
- Two 32-Byte Burstable FIFOs
- Active/Passive Add-On Bus Operation
- Mailbox Registers/w Byte Level Status
- Direct Mailbox Data Strobe/Int Pin
- Mailbox Read/Write Interrupts
- Direct PCI and Add-On Interrupt Pins
- Plug-N-Play Compatible
- Two-wire Serial Bus nvRAM Support
- Optional External BIOS capability
- 160-Pin PQFP

APPLICATIONS

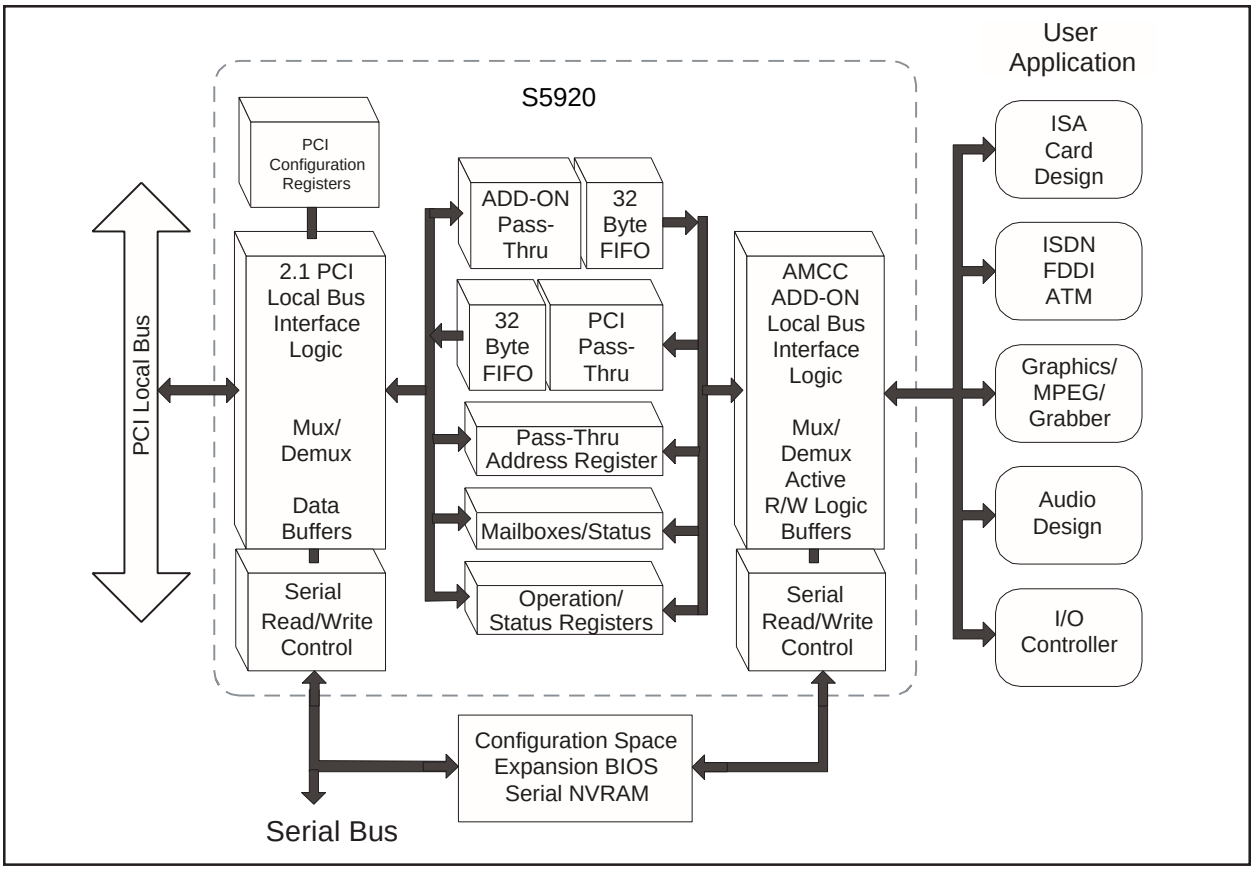
- ISA Conversions
- Multimedia
- I/O Ports
- Data Storage
- $CODEC_5$
- General Purpose PCI Bus Interfacing

ARCHITECTURAL OVERVIEW

The AMCC S5920 was developed to provide the designer with a single multi-function device offering a flexible and easy way to connect to the PCI bus. By using the S5920, the designer eliminates the task of assuring PCI bus specification compliance and the necessity of understanding PCI bus timing requirements when interfacing a new application.

The complex 33 MHz PCI bus signals are converted through the S5920 into an easy-to-use 8/16/32-bit user bus referred to as the user Add-On bus. The Add-On bus allows user add-on designs bus clock speed independent operation to 40 MHz.

Figure 1. S5920 Block Diagram



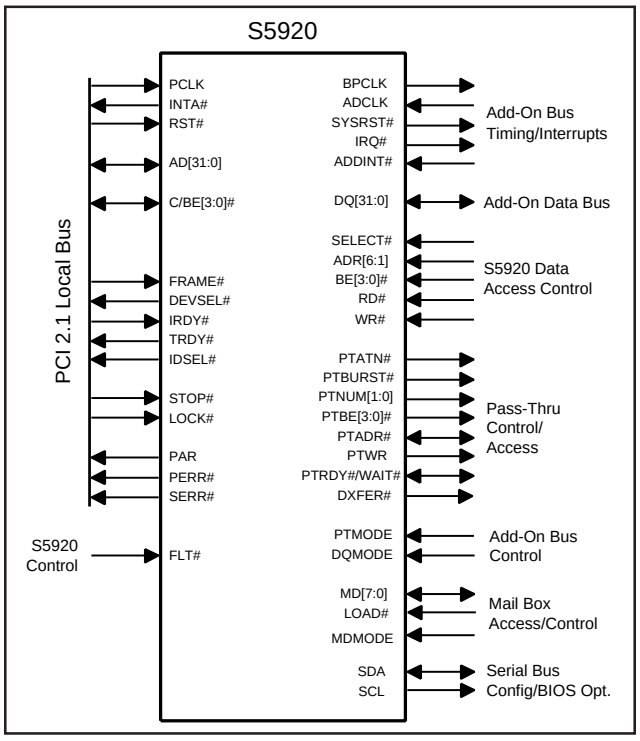
Since the S5920 is a PCI Target or Slave device only, its cost is significantly less than PCI Bus Master solutions. The S5920 is PCI purposed 2.2 compliant and can support data transfer rates up to 132 Mbytes/sec. Burst transfers and single data transfers are both supported. Figure 1 shows the block diagram for the S5920.

Many additional S5920 features offer the user easier hardware and software implementation. Up to four memory or I/O size definable blocks, referred to as Pass-Thru' regions, are provided for multiple device configurations. Data transfers via a Pass-Thru region can be performed either direct to the Add-On bus or through two 32-Byte burstable FIFOs. Added read prefetch and programmable FIFO wait state features allow the user to tune system performance. The Pass-Thru data channel also supports an active/passive mode bus interface. Passive mode requires the designer to transfer data by externally driving the Add-On bus. Active mode minimizes design components by enabling internal logic to drive or acquire the Add-On bus to read or write data independently. Active mode provides programmable wait state generation for slower Add-On designs.

Two 32-bit mailbox registers are implemented for additional data or user-defined status/command transfers. Each mailbox may be examined for empty or full, at the byte level, through a mailbox status register. Mailbox transfers can be either register style or hardware direct. Dedicated external mailbox data and strobe pins are provided for direct hardware read/writes and allow Add-On to PCI interrupt capabilities. A direct Add-On to a PCI bus interrupt pin is incorporated, adding design flexibility.

The S5920 supports a two-wire serial nvRAM. This allows the designer to customize the device configuration to be loaded during power-up initialization. An expansion BIOS may also be contained in the nvRAM.

Figure 2. S5920 Pinout



S5920 REGISTER ARCHITECTURE

S5920 communications, control and configuration is performed through three primary groups of registers: PCI Configuration Registers, PCI Operation Registers and Add-On Operation Registers. All of these registers are user configurable through their associated buses and from the external nvRAM. The following sections provide a brief overview of each register group and the nvRAM interface.

PCI Configuration Registers

All PCI compliant devices are required to provide a group of PCI configuration registers. These registers are polled by the host BIOS system during power-up initialization. They contain specific device and product information such as Vendor ID, Device ID, Subsystem Vendor ID, memory requirements, etc. These registers are located in the S5920 and are either initialized with predefined default values or user customized definitions contained in the external nvRAM.

PCI Bus Accessible Registers

The second group of registers are the PCI Operation Registers. This group of registers is accessible to the PCI Bus. These are the primary registers through which the PCI Host configures the S5920 operation and communicates with the Add-On Bus. These registers encompass the PCI bus mailboxes, Pass-Thru/ FIFO data channel and Status/ Control registers.

Add-On Bus Accessible Registers

The last register group consists of the Add-On Operation Registers. This group of registers is accessible via the Add-On Bus. These are the primary registers through which the Add-On application configures S5920 operation and communicates with the PCI Bus. These registers encompass the Add-On bus mailboxes, Pass-Thru/FIFO Registers and Status/Control Registers.

SERIAL NON-VOLATILE INTERFACE

Previously indicated, the S5920 contains the required set of PCI Configuration Registers. These registers can be initialized with default values or with customized values contained in an external nvRAM. The nvRAM allows Add-On card manufacturers to initialize the S5920 with their specific Vendor ID values, along with other desired S5920 operation characteristics.

MAILBOX OPERATION

The mailbox registers are divided into two 4-byte sets. Each set is dedicated to one bus for data transfer to the other bus. Figure 3 shows a block diagram of the mailbox section of the S5920. The provision of mailbox registers provides data or user defined command/status transfer capability between two buses. An empty/full indication for each mailbox register, at the byte level, is determined by polling a status register accessible to both the PCI and Add-On buses. Providing mailbox byte level full indications allows greater flexibility in 8-, 16-or 32-bit designs; i.e., transferring a single byte in 8-bit Add-On bus without requiring the assembly or disassembly of 32-bit data.

A mailbox byte level interrupt feature for PCI or Add-On buses is provided. Bit locations configured within the S5920 operation registers can select which mailbox byte is to generate an interrupt when the mailbox is written to. Interrupts can be generated to the PCI or Add-On buses. PCI bus interrupts may also be generated from direct hardware interfacing due to a unique S5920 feature. The Add-On mailbox is hardware accessible via a set of dedicated device pins. A single load pulse latches data into the mailbox generating an interrupt, if enabled.

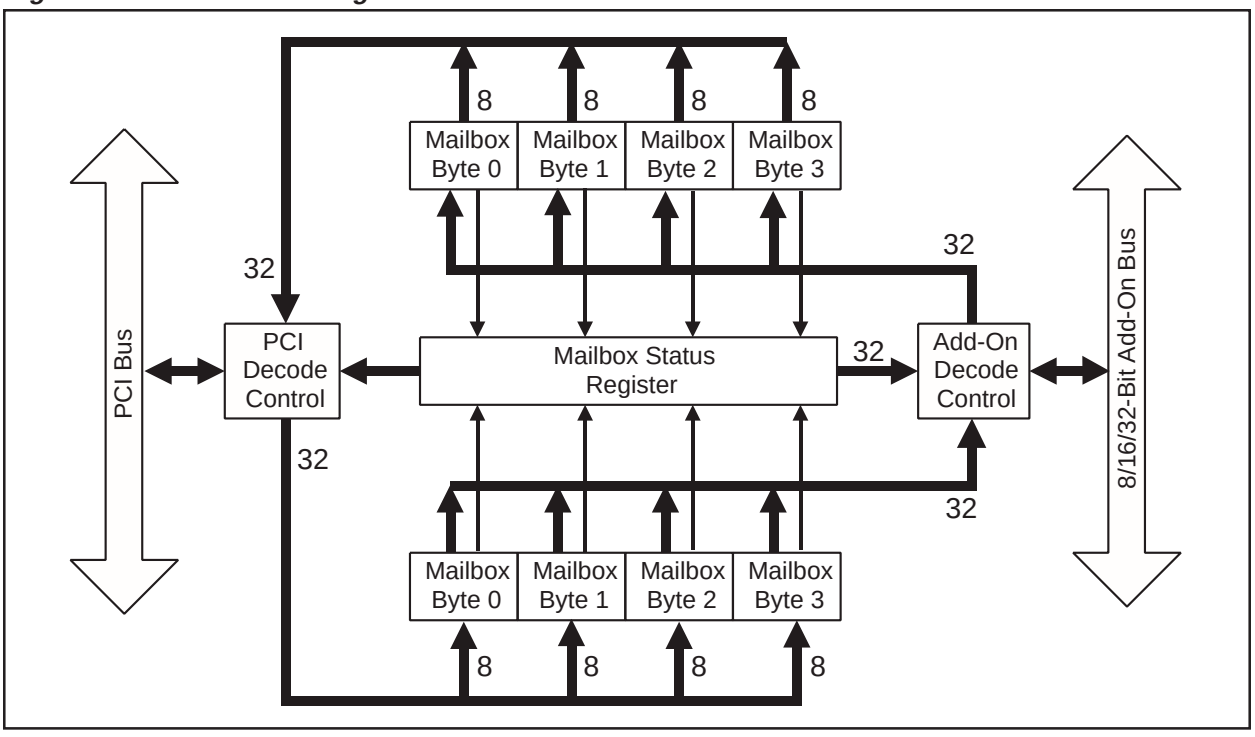
PASS-THRU OPERATION

Pass-Thru region accesses can execute PCI bus cycles in real time or through an internal FIFO. Real time operation allows the PCI bus to directly read or write to Add-On bus resources. The S5920 allows the designer to declare up to four individual Pass-Thru regions. Each region may be defined as 8, 16 or 32 bits wide, mapped into memory or I/O system space and may be up to 512 MB in size. Figure 4 shows a basic block diagram of the S5920 Pass-Thru architecture.

Host communications to the Pass-Thru data channel utilizes dedicated Add-On bus pins to signal that a PCI read or write has been requested. User logic decodes these signals to determine if it must read or write data to the S5920 to satisfy the PCI request. Information decoded includes: PCI read/write transaction request, the byte lanes involved, the specific Pass-Thru region accessed and whether the request is a burst or single cycle access.

Pass-Thru operation supports single PCI data cycles and PCI data bursts. During PCI burst operations, the S5920 is capable of transferring data at the full PCI bandwidth. Should slower Add-On logic be implemented, the S5920 will issue a PCI bus retry until the requested transfer is completed.

Figure 3. Mailbox Block Diagram

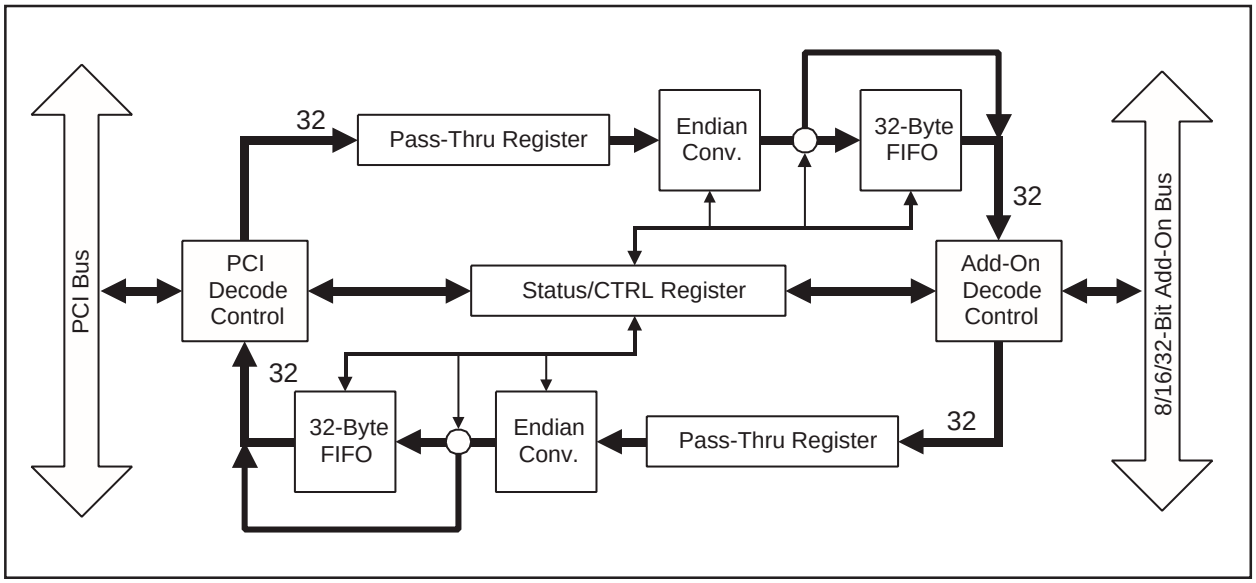


To increase data throughput, the Pass-Thru channel incorporates two 32-byte FIFOs. One FIFO is dedicated to PCI read data while the other is dedicated to PCI write data. Enabling the write FIFO allows the S5920 to accept zero wait state bursts from the PCI bus regardless of the Add-On bus application design speed. Figure 4 illustrates the Pass-Thru block.

Enabling the read FIFO allows data to be optionally prefetched from the Add-On Bus. This can greatly improve performance of slow Add-On bus designs. PCI read cycles can be performed with zero wait states since data has been prefetched into the FIFO. Either of the write/read FIFOs can be disabled or enabled to tune system performance.

The Add-On bus can be operated in two different modes: active or passive. The passive mode of operation mimics that of the S5933 Add-On bus operation. The user design drives S5920 pins to read or write data. In active mode, the Add-On bus is driven from an S5920 internal state machine. This reduces component count in cost-sensitive designs. Active mode also incorporates programmable wait states from 0 to 7.

Figure 4. Pass-Thru Block Diagram



Signal Type Definitions

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

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

Each signal that assumes the logic low state when asserted is followed by the pound sign (#). Example: TRDY# signal is asserted low when the target is ready to complete a data transfer. Signals that are not followed by the pound sign are asserted when they assume the logic high state.

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

A BYTE is an 8-bit object.

A WORD is a 16-bit, or 2-byte object.

A DWORD is a double word and is a 32-bit or 4-byte object.

All hex numbers are followed by an “h”. Examples:

9A4Fh

0110h

All binary numbers are followed by an “b”. Examples:

1010b

0110b

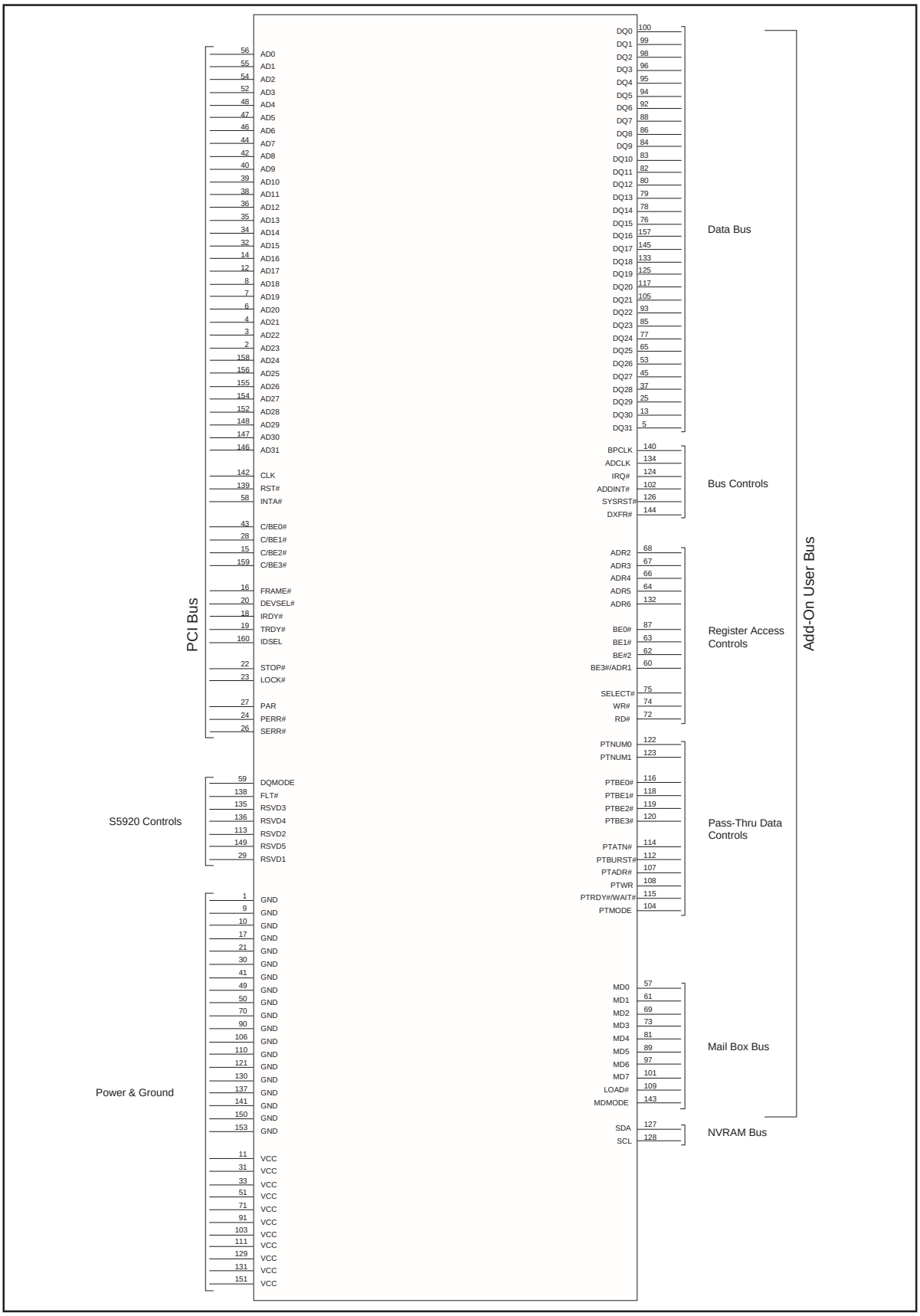
All decimal numbers are followed by an “d”. Examples:

4356d

1101d

Note: Tri-State is a Registered Trademark of National Semiconductor.

Figure 1. S5920 Pin Assignment



PCI BUS SIGNALS

The following sets of signals represent the interface pins available for the S5920 to PCI bus.

PCI Bus Address and Data Signal

Signal	Type	Description
AD[31:0]	t/s	Address/Data. Address and data are multiplexed on the same PCI bus pins. A PCI bus transaction consists of an address phase followed by one or more data phases. An address phase occurs on the PCLK cycle in which FRAME# is asserted. A data phase occurs on the PCLK cycles in which IRDY# and TRDY# are both asserted.
C/BE[3:0]#	in	Command/Byte Enable. Bus commands and byte enables are multiplexed on the same pins. These pins define the current bus command during an address phase. During a data phase, these pins are used as Byte Enables, with C/BE[0]# enabling byte 0 (LSB) and C/BE[3]# enabling byte 3 (MSB). **C/BE[3:0] Command Type** 0000 Interrupt Acknowledge 0001 Special Cycle 0010 I/O Read 0011 I/O Write 0100 Reserved 0101 Reserved 0110 Memory Read 0111 Memory Write 1000 Reserved 1001 Reserved 1010 Configuration Read 1011 Configuration Write 1100 Memory Read Multiple 1101 Dual Address Cycle 1110 Memory Read Line 1111 Memory Write and Invalidate
PAR	t/s	Parity. Parity is always driven as even from all AD[31:0] and C/BE[3:0]# signals. The parity is valid during the clock following the address phase and is driven by the bus master. During a data phase for write transactions, the bus master sources this signal on the clock following IRDY# active; during a data phase for read transactions, this signal is driven by the target and is valid on the clock following TRDY# active. The PAR signal has the same timing as AD[31:0], delayed by one clock.

PCI Bus System Signals

Signal	Type	Description
PCLK	in	PCI Clock. The rising edge of this signal is the reference upon which all other signals are based except for RST# and INTA#. The maximum PCLK frequency for the S5920 is 33 MHz and the minimum is DC (0 Hz).
RST#	in	Reset brings the S5920 to a known state: - All PCI bus output signals tri-stated. - All open drain signals (i.e., SERR#) floated. - All registers set to their factory defaults. - Pass-Thru is returned to an idle state. - All FIFOs emptied.

PCI Bus Data Transfer Control Signals

Signal	Type	Description
FRAME#	in	Frame. This signal is driven by the current bus master to indicate the beginning and duration of a bus transaction. When FRAME# is first asserted, it indicates a bus transaction is beginning with a valid addresses and bus command present on AD[31:0] and C/BE[3:0]. Data transfers continue while FRAME# is asserted. FRAME# de-assertion indicates the transaction is in a final data phase or has completed.
IRDY#	in	Initiator Ready. This signal is always driven by the bus master to indicate its ability to complete the current data phase. During write transactions, it indicates AD[31:0] contains valid data.
TRDY#	s/t/s	Target Ready. This signal is driven by the selected target to indicate the target is able to complete the current data phase. During read transactions, it indicates AD[31:0] contains valid data. Wait states occur until both TRDY# and IRDY# are asserted together.
STOP#	s/t/s	Stop. The Stop signal is driven by a selected target and conveys a request to the bus master to stop the current transaction.
LOCK#	in	Lock. The lock signal provides for the exclusive use of a resource. The S5920 may be locked by one master at a time.
IDSEL	in	Initialization Device Select. This pin is used as a chip select during configuration read or write transactions.
DEVSEL#	s/t/s	Device Select. This signal is driven by a target decoding and recognizing its bus address. This signal informs a bus master whether an agent has decoded a current bus cycle.
INTA#	o/d	Interrupt A. This signal is defined as optional and level sensitive. Driving it low will interrupt to the host. The INTA# interrupt is to be used for any single function device requiring an interrupt capability.

PCI Bus Error Reporting Signals

Signal	Type	Description
PERR#	s/t/s	Parity Error. Only for reporting data parity errors for all bus transactions except for Special Cycles. It is driven by the agent receiving data two clock cycles after the parity was detected as an error. This signal is driven inactive (high) for one clock cycle prior to returning to the tri-state condition.
SERR#	o/d	System Error. Used to report address and data parity errors on Special Cycle commands and any other error condition having a catastrophic system impact. Special Cycle commands are not supported by the S5920.

ADD-ON BUS AND S5920 CONTROL SIGNALS

The following sets of signals represent the interface signals available for the user Add-On bus and S5920 control.

Serial nvRAM Interface Signals

Signal	Type	Description
SCL	o/d-out	Serial Clock. This clock provides timing for all transactions on the two-wire serial bus. The S5920 drives this signal when performing as a serial bus master. SCL operates at the maximum allowable clock speed and enters the high Z state when FLT# is asserted or the serial bus is inactive.
SDA	o/d	Serial Data/Address. This bi-directional signal carries serial address and data information between nvRAMs and the S5920. This pin enters high Z state when FLT# is asserted or the serial bus is inactive.
Pin 135	in	Reserved. Must be left open.

Direct Mailbox Access Signals

Signal	Type	Description
MDMODE	in	Mailbox Data Mode. The MD[7:0] signal pins are always inputs when this signal is high. The MD[7:0] signal pins are defined as inputs and outputs under LOAD# control when MDMODE is low. This pin is provided for software compatibility with the S5933. New designs should permanently connect this signal low. This signal is connected to an internal pull-up.
LOAD#	in	MD[7:0] is defined as an input bus when this signal is low. The next rising edge of the ADCLK will latch MD[7:0] data into byte three of the Add-On outgoing mailbox. When LOAD# is high and MDMODE is low, MD[7:0] are defined as outputs displaying byte three of the PCI outgoing mailbox. This signal is connected to an internal pull-up.
MD[7:0]	t/s	Mailbox Data bus. The mailbox data registers can be directly accessed using the LOAD# and MDMODE signals. When configured as an input, data byte three of the PCI incoming mailbox is directly written to from these pins. When configured as an output, data byte three of the PCI outgoing mailbox is output to these pins. All MD[7:0] signals have an internal pull-up.

USER ADD-ON BUS PIN DESCRIPTIONS

The following sets of signals represent the interface pins available for the Add-On bus. The following defines three signal groups: S5920 register access signals, Pass-Thru channel signals, and general Add-On bus signals.

Pass-Thru Data Channel Pins

Signal	Type	Description
PTMODE	in	Pass-Thru Mode. Configures the Pass-Thru data channel operation. High configures the S5920 in Passive mode allowing other devices to read/write data bus data. Low configures the S5920 in Active mode. This mode allows the S5920 to actively drive signals and data onto the data bus. This signal is connected to an internal pull-up.
PTATN#	out	Pass-Thru Attention. Signals a decoded PCI to Pass-Thru region bus cycle. PTATN# is generated to signal that Add-On logic Pass-Thru data must be read from or written to the S5920.
PTBURST#	out	Pass-Thru Burst. Informs the Add-On bus that the current Pass-Thru region decoded PCI bus cycle is a burst access.
PTRDY#/WAIT#	in	Pass-Thru Ready/Pass-Thru Wait. During passive mode, the signal is referred to as PTRDY# and is asserted low to indicate Add-On logic has read/written data in response to a PTATN# signal. During active mode operation, the signal is referred to as WAIT# and can be driven low to insert wait states or hold the S5920 from clocking data onto the data bus. PTRDY# or WAIT# is synchronous to ADCLK.
PTNUM[1:0]	out	Pass-Thru Number. Identifies which of the four Pass-Thru regions the PTATN# read/write is requesting. Only valid for the duration of PTATN# active. 00 = Base Address Register 1, 01 = Base Address Register 2, 10 = Base Address Register 3, 11 = Base Address Register 4.
PTBE[3:0]#	out	Pass-Thru Byte Enables. During a PCI to Pass-Thru read, PTBE[3:0] indicate which bytes of a DWORD are to be written into. During a PCI to Pass-Thru write, these pins indicate which bytes of a DWORD are valid to read. PTBE[3:0]# are only valid while PTATN# is asserted.
PTADR#	t/s	Pass-Thru Address. Is an input when in passive mode. When asserted, the 32-bit Pass-Thru address register contents are driven onto the DQ[31:0] bus. All other Add-On control signals must be inactive during the assertion of PTADR# in passive mode. In active mode, becomes an output and indicates a Pass-Thru address is on the DQ bus. The DQMODE signal does not affect DQ bus width while the Pass-Thru address is driven.
PTWR	out	Pass-Thru Write. This signal indicates whether the current PCI to Pass-Thru bus transaction is a read or write cycle. Valid only when PTATN# is active.
DXFER#	out	ACTIVE Transfer complete. When in ACTIVE mode, this output is asserted at the end of every 8- 16- or 32-bit data transfer cycle. This signal is not used in Passive mode.

S5920 Add-On Bus Register Access Pins

Signal	Type	Description
DQ[31:0]	t/s	Address/Data bus. The 32-bit Add-On data bus. The DQMODE signal configures the bus width for either 32 or 16 bits. All DQ[31:0] signals have an internal pull-up.
ADR[6:2]	in	Address [6:2]. These inputs select which S5920 register is to be read from or written to. To be used in conjunction with SELECT#, BE[3:0]# and WR# or RD#. The register addresses are as follows: **ADR[6:2] Register Name** 0 0 0 1 1 Add-On Incoming Mailbox Register 0 0 1 1 1 Add-On Outgoing Mailbox Register 0 1 0 1 0 Add-On Pass-Thru Address Register 0 1 0 1 1 Add-On Pass-Thru Data Register 0 1 1 0 1 Add-On Mailbox Status Register 0 1 1 1 0 Add-On Interrupt Control Register 0 1 1 1 1 Add-On Reset ControlRegister 1 0 0 0 0 Pass-Thru/FIFO Configuration Register Note: ADR[6:2] bits begin at bit position two. All references to an address, in hex, adds bits 0 and 1 as zeros. Example: The Add-On incoming mailbox register is referenced as 0Ch.
BE[2:0]#	in	Byte Enable 2 through 0. Provides individual read/write byte enabling during register read or write transactions. BE2# enables activity over DQ[23:16], BE1# enables DQ[15:8], and BE0# enables DQ[7:0]. During read transactions, these pins enable the output driver for each byte lane; for write transactions, they serve as an input enable to perform the write to each byte lane.
BE3# / ADR1	in	Byte Enable [3] for a 32-bit bus width / Address [1] for a 16-bit bus width. BE3#, enables DQ[31:24] input drivers for writing data to registers identified by ADR[6:2] and enables DQ[31:24] output drivers to read registers identified by ADR[6:2]. To be used in conjunction with SELECT# and RD# or WR#. ADR1, selects the upper or lower WORD of a DWORD when a 16-bit-wide bus is selected. 1 = upper, 0 = lower.
SELECT#	in	Select. Enables internal S5920 logic to decode WR#, RD# and ADR[6:2] when reading or writing to any Add-On register.
WR#	in	Write Enable. Asserting this signal writes DQ bus data byte(s) selected by BE[3:0]# into the S5920 register defined by SELECT# and ADR[6:2].
RD#	in	Read Enable. Asserting this signal drives data byte(s) selected by BE[3:0]# from the S5920 register defined by SELECT# and ADR[6:2] onto the DQ bus.
DQMODE	in	DQ Mode. Defines the DQ bus width when accessing data using WR#, RD#, SELECT# and ADR[6:2]#. Low = 32-bit wide DQ bus. High = 16-bit wide DQ bus. When high, the signal BE3# is re-assigned to the ADR1 signal and only DQ[15:0] is active. Note: This pin only affects DQ Bus Width for S5920 Data Registers. This pin has no effect on accesses DQ Bus Width. For the Pass-Thru data register (APTD, ADR = 2Ch). The width of the DQ bus is determined by the region-size bits in the corresponding Base Address Register. In addition, DQMODE has no effect when using the direct-access pin PTADR#. When PTADR# is asserted, all 32 bits of the Pass-Thru address are provided.

Add-On Bus General Pins

Signal	Type	Description
SYSRST#	out	System Reset. An active-low buffered PCI bus RST# output signal. The signal is asynchronous and can be asserted through software from the PCI host interface.
BPCLK	out	Buffered PCI Clock. This output is a buffered form of the PCI bus clock and has all of the behavioral characteristics of the PCI clock (i.e., DC-to-33 MHz capability).
ADCLK	in	Add-On Clock. All internal S5920 Add-On bus logic is synchronous to this clock. The clock is asynchronous to the PCI bus logic unless connected to the BPCLK signal.
IRQ#	out	Interrupt Request. This output signals to Add-On logic that a significant event has occurred as a result of activity within the S5920.
ADDINT#	in	Add-On interrupt. When enabled and asserted, this input will cause a PCI bus interrupt by driving INTA# low. The input is level sensitive and can be driven by multiple sources. This signal is connected to an internal pull-up.
FLT#	in	Float. Floats all S5920 output signals when asserted. This signal is connected to an internal pull-up
Pin 149	X	For factory use only. Must be left open.
Pin 136	X	For factory use only. Must be left open.
Pin 135	X	For factory use only. Must be left open.
Pin 113	X	For factory use only. Must be left open.
Pin 29	X	For factory use only. Must be left open.

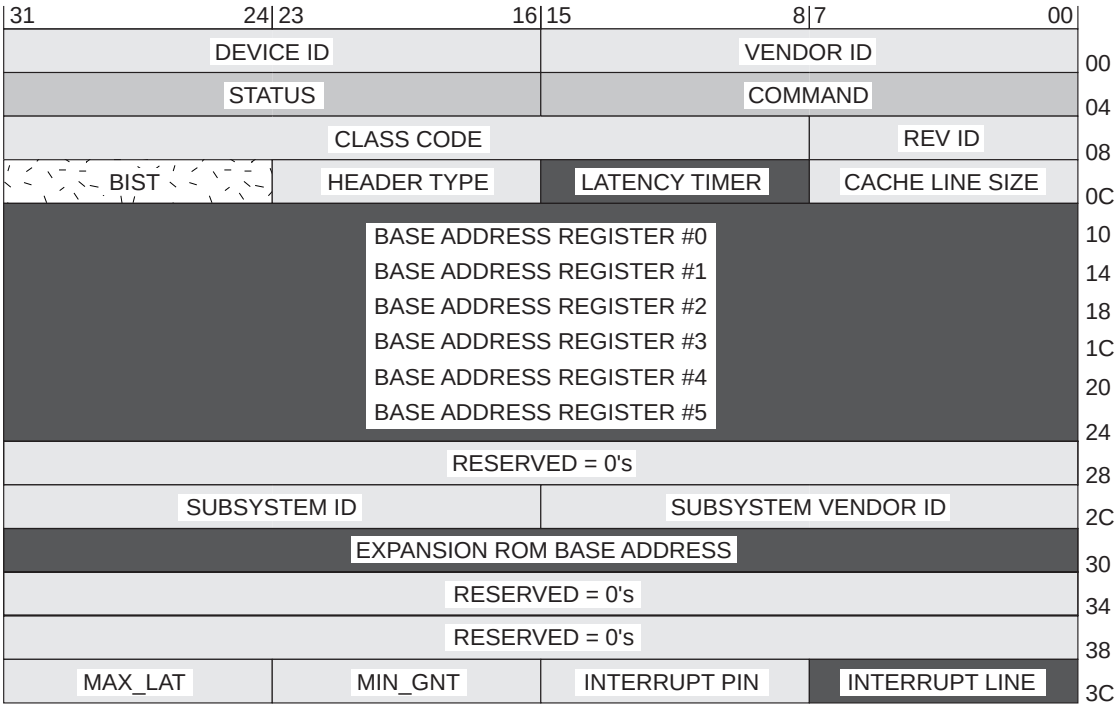
Each PCI bus device contains a unique 256-byte region called its configuration header space. Portions of this configuration header are mandatory in order for a PCI agent to be in full compliance with the PCI specification. This section describes each of the configuration space fields—its address, default values, initialization options, and bit definitions—and also provides an explanation of its intended usage.

Table 1. Configuration Registers

Address Offset	Abbreviation	Register Name
00h–01h	VID	Vendor Identification Register
02h–03h	DID	Device Identification Register
04h–05h	PCICMD	Command Register
06h–07h	PCISTS	Status Register
8h0	RID	Revision Identification Register
09h–0Bh	CLCD	Class Code Register
0Ch	CALN	Cache Line Size Register
0Dh	LAT	Latency Timer Register
0Eh	HDR	Header Type Register
0Fh	BIST	Built-in Self-test Register
10h–27h	BADR0-BADR5	Base Address Registers (0-5)[1]
28h–2Bh	—	Reserved
2Ch–2Dh	SVID	Subsystem Vendor Identification Register
2Eh–2Fh	SID	Subsystem Identification Register
30h–33h	XROM	Expansion ROM Base Address Register
34h–3Bh	—	Reserved
3Ch	INTLN	Interrupt Line Register
3Dh	INTPIN	Interrupt Pin Register
3Eh	MINGNT	Minimum Grant Register
3Fh	MAXLAT	Maximum Latency Register
40h–FFh	—	Not used

1. BADR 5 is not implemented in the S5920.

PCI Configuration Space Header



LEGEND

EPROM IS DATA SOURCE (READ ONLY)

CONTROL FUNCTION

EPROM INITIALIZED RAM (CAN BE ALTERED FROM PCI PORT)

EPROM INITIALIZED RAM (CAN BE ALTERED FROM ADD-ON PORT)

HARD-WIRED TO ZEROES

Note: Some registers are a combination of the above. See individual sections for full description.

VENDOR IDENTIFICATION REGISTER (VID)

Register Name: Vendor Identification

Address Offset: 00h-01h

Power-up value: 10E8h (AMCC's)

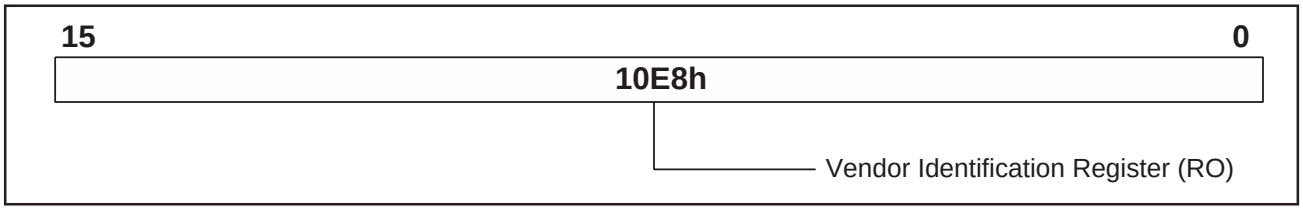
Boot-load: External nvRAM offset 040h-41h

Attribute: Read Only

Size: 16 Bits

This register is reserved for the manufacturer's device identification number (VID). VID numbers are assigned to each PCI device manufacturer by an international organization known as the PCI Special Interest Group (SIG). This ensures PCI device uniqueness among all manufacturers. This register defaults to AMCC's VID during power-on initialization. The default value can be changed to another valid VID when an external non-volatile device is used for boot loading.

Figure 1. Vendor Identification Register



Bit	Description
15.0	Vendor Identification Number: AMCC's 16-bit value is 10E8h

DEVICE IDENTIFICATION REGISTER (DID)

Register Name: Device Identification

Address Offset: 02h-03h

Power-up value: 5920h

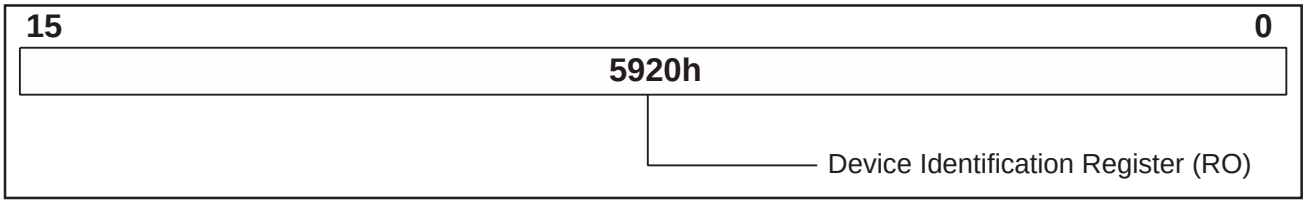
Boot-load: External nvRAM offset 042h-43h

Attribute: Read Only

Size: 16 bits

This register is reserved for the manufacturer's device identification number (DID). DID numbers are maintained and issued by the registered owner of the VID number programmed into a PCI device. AMCC will issue a DID number to manufacturers using the AMCC VID. This maintains PCI device uniqueness among all manufacturers. to be in compliance. This register defaults to AMCC's DID during power-on initialization. The default value can be changed to another valid DID when an external non-volatile device is used for boot loading.

Figure 2. Device Identification Register



Bit	Description
15.0	Device Identification Number: AMCC's temporary end user value 5920h

PCI COMMAND REGISTER (PCICMD)

Register Name: PCI Command

Address Offset: 04h-05h

Power-up value: 0000h

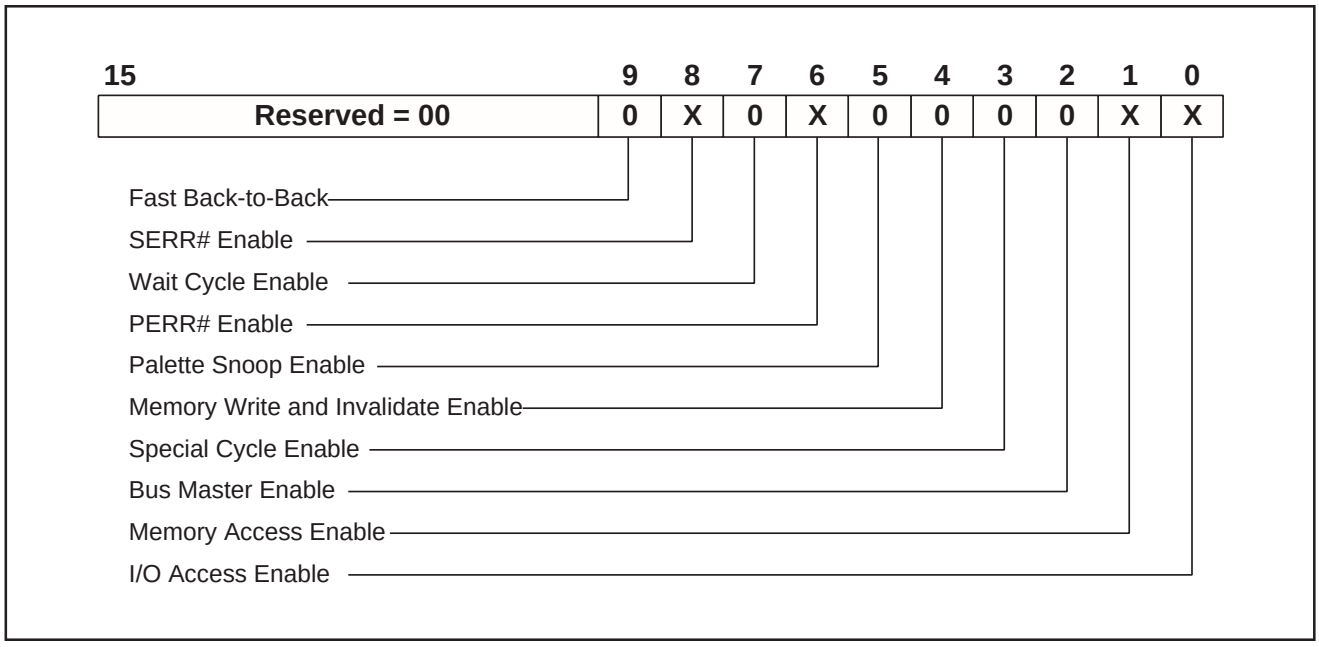
Boot-load: not used

Attribute: Read/Write (R/W on 4 bits, R/O for all others)

Size: 16 bits

This 16-bit register provides basic control over a device's ability to respond to or perform PCI accesses. This register is defined by the PCI specification and its implementation is required of all PCI devices. Four of the ten implemented bits are required by the S5920; those which are not required are hardwired to 0. The definitions for all the fields are provided here for completeness.

Figure 3. PCI Command Register



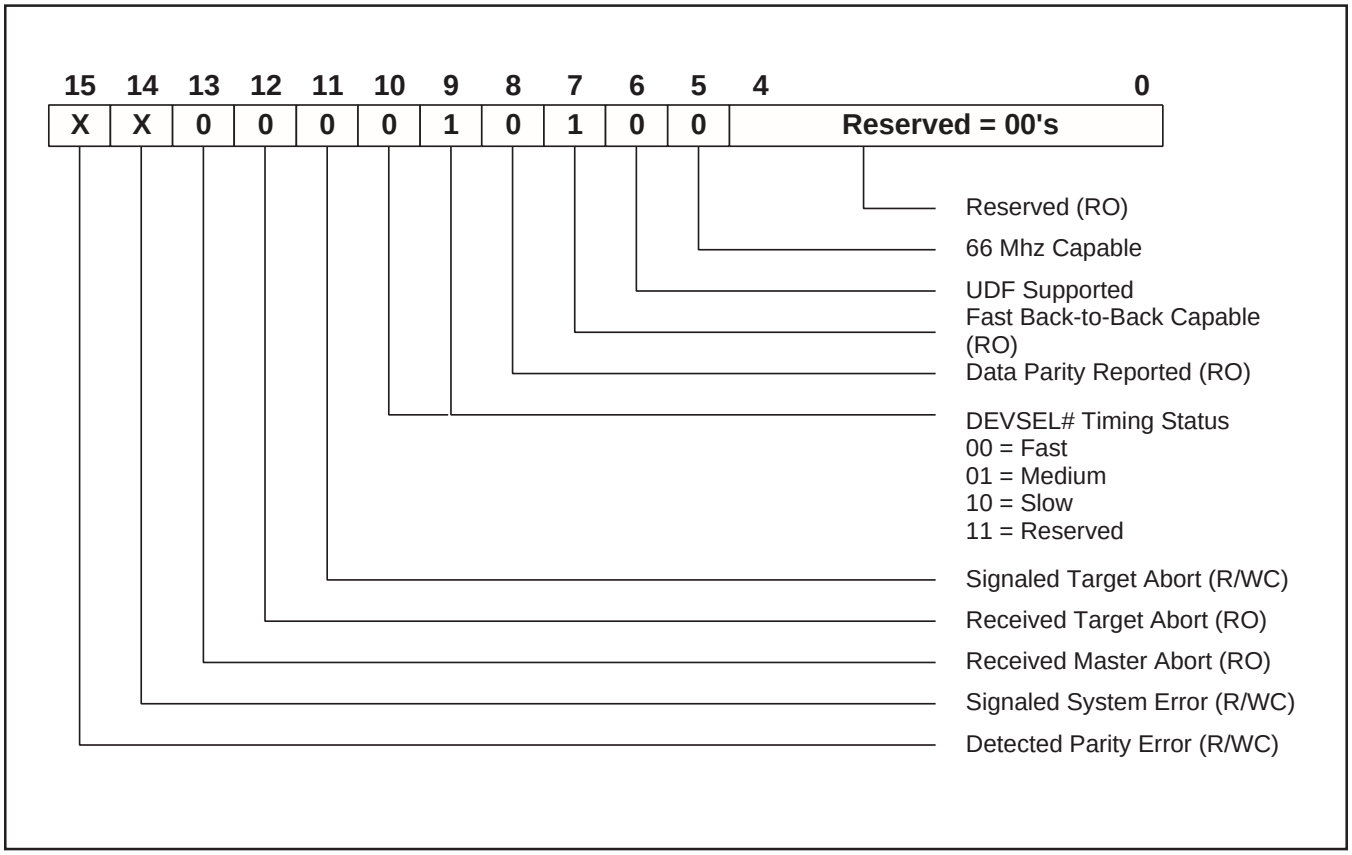
Bit	Description
15:10	Reserved. Hardwired to 0.
9	Fast Back-to-Back Enable. This bit enables fast back-to-back capability for bus master transaction. The S5920 is a target-only device and hardwires this bit to a 0.
8	System Error Enable. Setting this bit to a 1 allows the S5920 to drive the SERR# signal. Setting to a 0 will disable the output driver. The assertion of RESET# will set this bit to a 0. The SERR# pin driven active normally signifies a parity error occurred during a PCI address phase.
7	Wait Cycle Enable. Controls whether a device implements address/data stepping. This bit is hardwired to 0 as the S5920 does not uses stepping.
6	Parity Error Enable. This bit allows the S5920 to drive the PERR# and to generate a SERR# signal. A one allows the parity generation and a 0 will disable generation of a parity error indication. This bit is set to 0 when RESET# is asserted.
5	Palette Snoop Enable. Enables VGA compatible devices to perform palette snooping. This bit is hardwired to a 0 as the S5920 is not a PCI-based VGA device.
4	Memory Write and Invalidate Enable. This bit enables bus masters to generate Memory Write and Invalidate PCI bus commands when set to a 1. When set to 0, bus masters generate memory write commands instead. The S5920 is a PCI target only and therefore hardwires this bit to 0.
3	Special Cycle Enable. Setting this bit to one enables devices monitoring of PCI special cycles. The S5920 does not monitor (or generate) special cycles and hardwires this bit to 0.
2	Bus Master Enable. This bit allows a PCI device to function as a Bus Master. The S5920 is a PCI target device only and hardwires his bit to 0.
1	Memory Space Enable. This bit enables S5920 memory region decodes to any of the five defined base address register memory regions and the Expansion ROM Base Address Register. This bit is cleared to 0 when RESET# is asserted.
0	I/O Space Enable. This bit enables S5920 I/O region decodes to any of the five defined base address register I/O regions. This bit is cleared to 0 when RESET# is asserted.

PCI STATUS REGISTER (PCISTS)

Register Name:	PCI Status
Address Offset:	06h-07h
Power-up value:	0200h
Boot-load:	not used
Attribute:	Read Only
	Read/Write Clear
Size:	16 bits

This register contains PCI device status information. This register is defined by the PCI specification and its implementation is required of all PCI devices. Only applicable bits are used by the S5920; those which are not used are hardwired to 0. Status bits within this register are designated as “write one clear,” meaning that in order to clear a given bit, a 1 must be written. All R/W/C bits written with a 0 are left unchanged. These bits are identified in Figure 4 as (R/WC). Those which are Read Only are shown as (RO).

Figure 4. PCI Status Register



Bit	Description
15	Detected Parity Error. This bit is set whenever the S5920 detects a parity error. It is set independent of the state of Command Register Bit 6. The bit is cleared by writing a 1.
14	Signaled System Error. This bit is set whenever the S5920 generates the SERR# signal. This bit can be reset by writing a 1.
13	Received Master Abort. Bus master devices set this bit to indicate a bus master transaction has been terminated due to a master abort. The S5920 is a target device and hardwires this to 0.
12	Received Target Abort. This bit is set by a bus master when its transaction is terminated by a target abort from the currently addressed target device. This bit is required for bus masters and is hardwired to 0 in the S5920.
11	Signaled Target Abort. This bit is set the target device whenever it terminates a transaction with a target abort. The S5920 does not issue target aborts and hardwires this bit to 0.
10:9	Device Select Timing. These bits are read-only and define the DEVSEL# timing for a target device. The S5920 is a medium PCI device.
8	Data Parity Reported. Only implemented by bus mastering devices to notify a parity error has been detected. This is not applicable to the S5920 and is hardwired to 0.
7	Fast Back-to-back Capable. This read-only bit indicates if a target device supports fast back-to-back transactions. The S5920 supports this feature and hardwires the bit to 1.
6	UDF Supported. 1 = device supports user-definable features. 0 = device does not support user-definable features. The S5920 implements definable memory regions and hardwires this bit to 0.
5	66 MHz Capable. 1 = device is capable of running at 66 MHz. 0 = device is capable of running at 33 MHz. This bit is hardwired to 0.
4:0	Reserved. Hardwired to zero.

REVISION IDENTIFICATION REGISTER (RID)

Register Name: Revision Identification

Address Offset: 08h

Power-up value: 00h

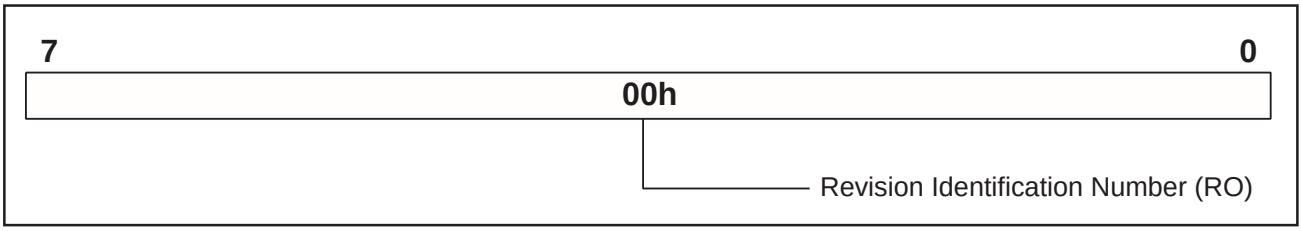
Boot-load: External nvRAM/EPROM offset 048h

Attribute: R/W

Size: 8 Bits

This register is reserved for AMCC's S5920 silicon revision identification number. The register defaults to that value after power up. Write operations from the PCI interface have no effect on the register. User-defined values can be boot-loaded from an optional external non-volatile. AMCC does not recommend changing the register value. The Subsystem Vendor ID and/or Subsystem ID are intended for end user information.

Figure 5. Revision Identification Register



Bit	Description
7:0	Revision Identification Number: Initialized to the S5920 silicon revision.

CLASS CODE REGISTER (CLCD)

Register Name:	Class Code
Address Offset:	09h-0Bh
Power-up value:	FF0000h
Boot-load:	External nvRAM offset 049h-4Bh
Attribute:	Read Only
Size:	24 Bits

This 24-bit, read-only register is divided into three one-byte fields: the base class byte at location 0Bh, the sub-class byte at 0Ah, and the programming interface byte at 09h. The default setting for the base class is FFh, which indicates that the device does not fit into the thirteen base classes defined in the PCI Bus Specification. It is possible, however, through use of the external non-volatile memory to change the value of this register. Refer to the PCI specification for details.

Figure 6. Class Code Register

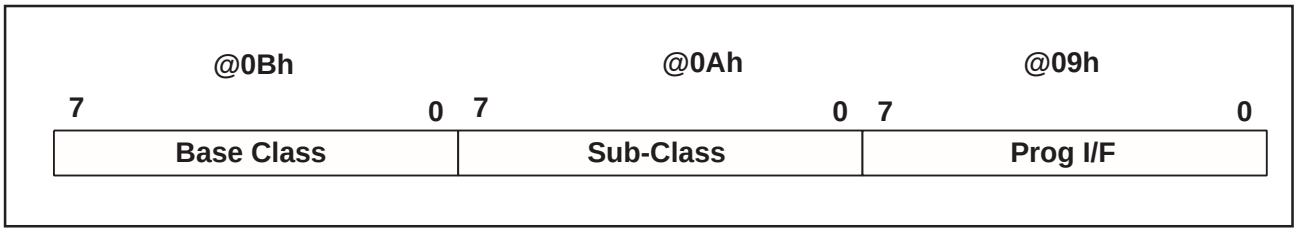


Table 2. Defined Base Class Codes

Base-Class	Description
00h	Early, pre-2.0 PCI specification devices
01h	Mass storage controller
02h	Network controller
03h	Display controllers
04h	Multimedia devices
05h	Memory controllers
06h	Bridge devices
07h	Simple communication controllers
08h	Generic system peripherals
09h	Input devices
0Ah	Docking stations
0Bh	Processors
0Ch	Serial bus controllers
0D-FEh	Reserved
FFh	Device does not fit defined class codes (default)

Table 3. Base Class Code 00h: Early, Pre-2.0 Specification Devices

Sub-Class	Prog I/F	Description
00h	00h	All currently implemented devices except VGA-compatible devices
01h	00h	VGA-compatible devices

Table 4. Base Class Code 01h: Mass Storage Controllers

Sub-Class	Prog I/F	Description
00h	00h	SCSI controller
01h	xxh	IDE controller
02h	00h	Floppy disk controller
03h	00h	IPI controller
04h	00h	RAID controller
80h	00h	Other mass storage controller

Table 5. Base Class Code 02h: Network Controllers

Sub-Class	Prog I/F	Description
00h	00h	Ethernet controller
01h	00h	Token ring controller
02h	00h	FDDI controller
03h	00h	ATM controller
80h	00h	Other network controller

Table 6. Base Class Code 03h: Display Controllers

Sub-Class	Prog I/F	Description
00h	00h	VGA-compatible controller
00h	01h	8514 compatible controller
01h	00h	XGA controller
80h	00h	Other display controller

Table 7. Base Class Code 04h: Multimedia Devices

Sub-Class	Prog I/F	Description
00h	00h	Video device
01h	00h	Audio device
80h	00h	Other multimedia device

Table 8. Base Class Code 05h: Memory Controllers

Sub-Class	Prog I/F	Description
00h	00h	RAM memory controller
01h	00h	Flash memory controller
80h	00h	Other memory controller

Table 9. Base Class Code 06h: Bridge Devices

Sub-Class	Prog I/F	Description
00h	00h	Host/PCI bridge
01h	00h	PCI/ISA bridge
02h	00h	PCI/EISA bridge
03h	00h	PCI/Micro Channel bridge
04h	00h	PCI/PCI bridge
05h	00h	PCI/PCMCIA bridge
06h	00h	NuBus bridge
07h	00h	CardBus bridge
80h	00h	Other bridge type

Table 10. Base Class Code 07h: Simple Communications Controllers

Sub-Class	Prog I/F	Description
00h	00h	Generic XT compatible serial controller
	01h	16450 compatible serial controller
	02h	16550 compatible serial controller
01h	00h	Parallel port
	01h	Bidirectional parallel port
	02h	ECP 1.X compliant parallel port
80h	00h	Other communications device

Table 11. Base Class Code 08h: Base System Peripherals

Sub-Class	Prog I/F	Description
00h	00h	Generic 8259 PIC
	01h	ISA PIC
	02h	EISA PIC
01h	00h	Generic 8237 DMA controller
	01h	ISA DMA controller
	02h	EISA DMA controller
02h	00h	Generic 8254 system timer
	01h	ISA system timer
	02h	EISA system timers (2 timers)
03h	00h	Generic RTC controller
	01h	ISA RTC controller
80h	00h	Other system peripheral

Table 12. Base Class Code 09h: Input Devices

Sub-Class	Prog I/F	Description
00h	00h	Keyboard controller
01h	00h	Digitizer (Pen)
02h	00h	Mouse controller
80h	00h	Other input controller

Table 13. Base Class Code 0Ah: Docking Stations

Sub-Class	Prog I/F	Description
00h	00h	Generic docking station
80h	00h	Other type of docking station

Table 14. Base Class Code 0Bh: Processors

Sub-Class	Prog I/F	Description
00h	00h	Intel386™
01h	00h	Intel486™
02h	00h	Pentium™
10h	00h	Alpha™
40h	00h	Co-processor

Table 15. Base Class Code 0Ch: Serial Bus Controllers

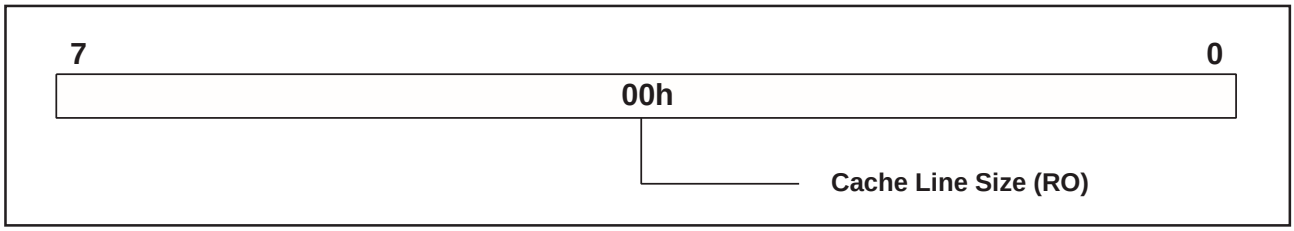
Sub-Class	Prog I/F	Description
00	00h	FireWire™ (IEEE 1394)
01h	00h	ACCESS.bus
02h	00h	SSA
03h	00h	Universal Serial Bus
04h	00h	Fibre Channel

CACHE LINE SIZE REGISTER (CALN)

Register Name:	Cache Line Size
Address Offset:	0Ch
Power-up value:	00h, hardwired
Boot-load:	not used
Attribute:	Read Only
Size:	8 bits

The cache line configuration register is used by bus masters implementing memory write and invalidate commands. The register defines the cache line size in double word (64-bit) increments. The S5920 is a target device not requiring cache. The register is hardwired to 0.

Figure 7. Cache Line Size Register



LATENCY TIMER REGISTER (LAT)

Register Name: Latency Timer

Address Offset: 0Dh

Power-up value: 00h

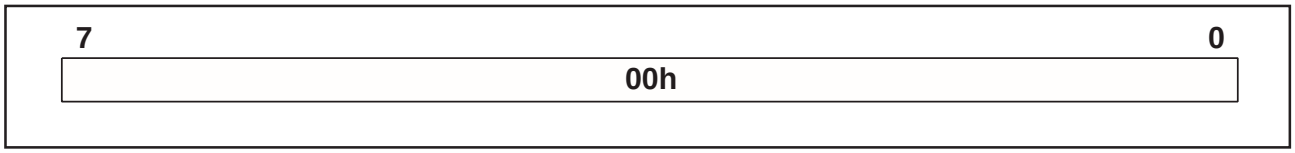
Boot-load: not used

Attribute: Read Only

Size: 8 bits

The latency timer defines the minimum amount of time that a bus master can retain ownership of the PCI bus. The S5920 is a target device requiring zero bus ownership time. The register is hardwired to zero.

Figure 8. Latency Timer Register

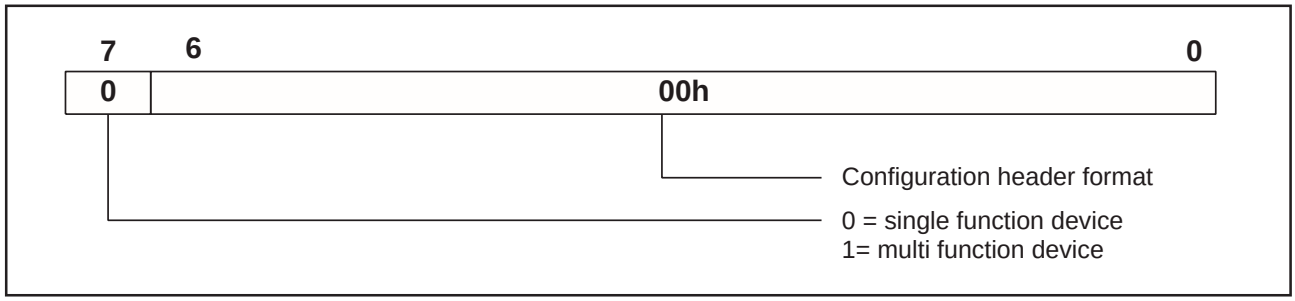


HEADER TYPE REGISTER (HDR)

Register Name:	Header Type
Address Offset	0Eh
Power-up value:	00h, Hardwired
Boot-load:	External nvRAM offset 04Eh
Attribute:	Read Only
Size:	8 bits

This register consists of two fields: Bits 6:0 define the format of double words 4d through 15d of the device's configuration header. Bit 7 defines whether the device is a single function or a multi-function PCI bus agent. The S5920 is defined as a single function PCI device.

Figure 9. Header Type Register

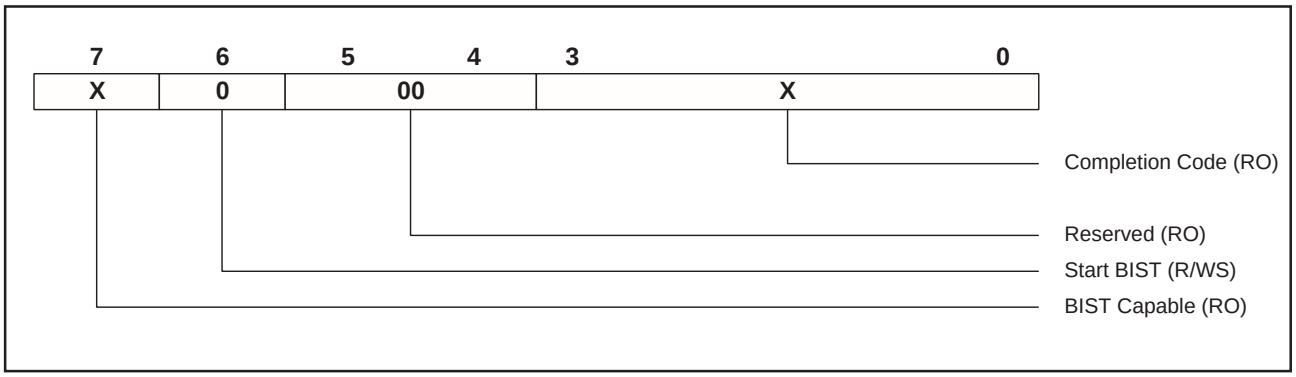


BUILT-IN SELF-TEST REGISTER (BIST)

Register Name:	Built-in Self-Test
Address Offset	0Fh
Power-up value:	00h
Boot-load:	External nvRAM/EPROM offset 04Fh
Attribute:	D7, D5-0 Read Only, D6 as PCI bus write only
Size:	8 bits

The Built-In Self-Test (BIST) Register permits the implementation of custom, user-specific diagnostics. This register has four fields shown in Figure 10. Bit 7 defines S5920's support of a built-in self test. When bit 7 is set, writing a 1 to bit 6 produce an interrupt signal on the Add-On bus. Bit 6 remains set until cleared by a write operation to this register from the Add-On bus interface. When bit 6 is reset, it is interpreted as completion of the self-test and an error is indicated by a non-zero value for the completion code (bits 3:0).

Figure 10. Built-In Self-Test Register



Bit	Description
7	BIST Capable. This bit indicates the Add-On device supports a built-in self-test when a 1 is returned. A 0 should be returned if this self-test feature is not required. This field is read only from the PCI interface.
6	Start BIST. Writing a 1 to this bit indicates that the self-test should start. This bit can only be written if bit 7 is one. When bit 6 is set, an interrupt is issued to the Add-On interface. Other than through a reset, Bit 6 can only be cleared by a write to this element from the Add-On bus interface. The PCI bus specification requires that this bit be cleared within 2 seconds after being set, or the device will be failed. This bit is read/write set (R/WS).
5:4	Reserved. These bits are reserved and are hardwired to 0.
3:0	Completion Code. This field provides a method for detailing a device-specific error. It is considered valid when the start BIST (bit 6) changes from 1 to 0. An all-zero value for the completion code indicates successful completion.

BASE ADDRESS REGISTER (BADR)

Register Name:	Base Address
Address Offset:	10h, 14h, 18h, 1Ch, 20h
Power-up value:	FFFFFF81h for offset 10h; 00000000h for all others
Boot-load:	External nvRAM offset 050h, 54h, 58h, 5Ch, 60h (BADR0-4)
Attribute:	high bits Read/Write; low bits Read Only
Size:	32 bits

Base address registers are used by the system BIOS to determine how much memory or I/O address space a region requires in host space. The actual memory or I/O location(s) of the space is determined by interrogating these registers after BIOS power-up initialization. Bit zero of each field is used to select whether the space required is to be decoded as memory (bit 0 = 0) or I/O (bit 0 = 1). Since this PCI device has internal operating registers, the Base Address Register at offset 10h is assigned to them. The remaining four base address registers can only be used by boot-loading them from the external nvRAM interface.

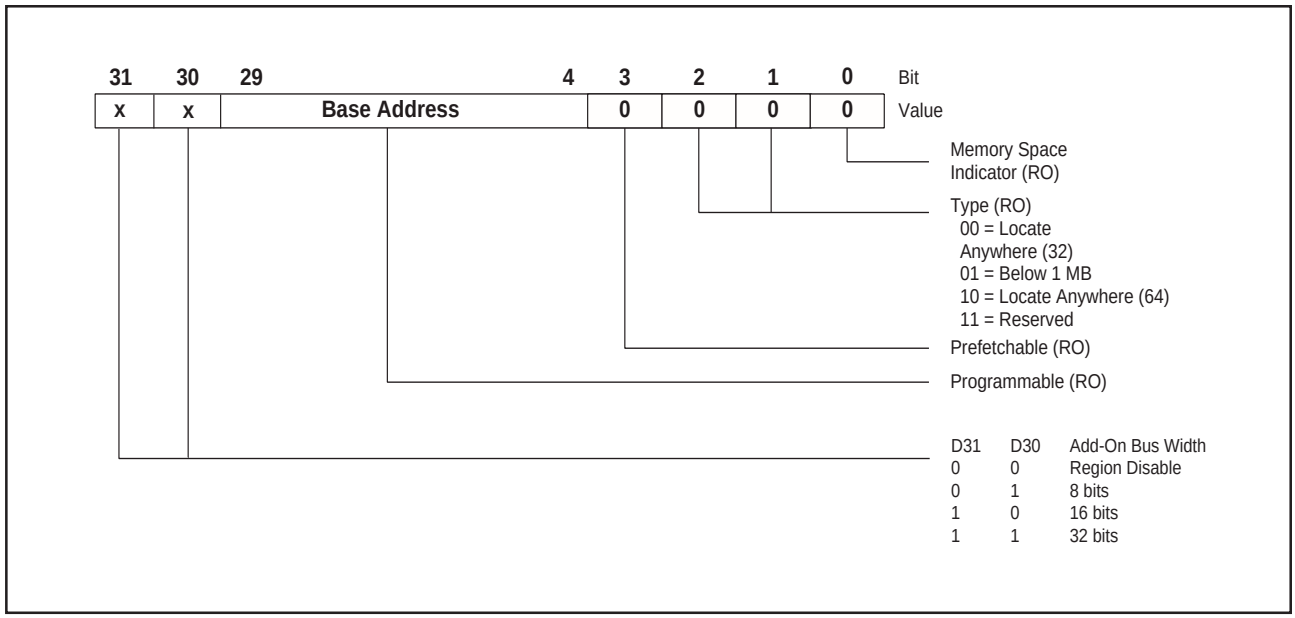
Determining Base Address Size

The address space defined by a given base address register is determined by writing all 1s to a given base address register from the PCI bus and then reading that register back. The number of 0s returned starting from D4 for memory space and D2 for I/O space toward the high-order bits reveals the amount of address space desired. Tables 17 and 18 list the possible returned values and their corresponding size for both memory and I/O, respectively. Included in the tables are the nvRAM/EPROM boot values which correspond to a given assigned size. A register returning all 0s indicates the region is disabled.

Assigning the Base Address

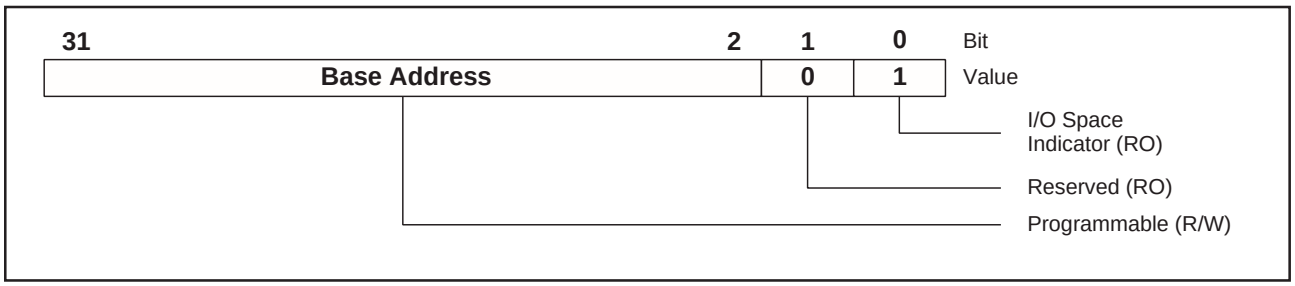
After a base address has been sized, the BIOS can physically locate it in memory (or I/O) space. The base address value must be on a natural binary boundary for the required size. For example, the first base address register returns FFFFFF81h indicating an I/O space (D0=1) of size 80h. This means that the 5920's internal registers can be selected for I/O addresses between 00000300h through 0000037Fh, in this example. (example 300h, 380h etc.; 338h, 340h would not be allowable).

Figure 11a. Base Address Register - Memory



Bit	Description
31:4	Base Address Location. These bits locate the decoded region in memory space. Only bits which return a 1 after being written as 1 are usable for this purpose. Except for Base Address Register 0, these bits are individually enabled by the contents sourced from the external boot memory.
3	Prefetchable. When set as a 1, this bit signifies that this region of memory can be cached. Cacheable regions can only be located within the region altered through PCI bus memory writes. This bit, when set, also implies that all read operations will return the data associated for all bytes regardless of the Byte Enables. Memory space which cannot support this behavior should leave this bit in the zero state. this bit is set by the reset pin and later initialized by the external boot memory option. Base Address Register 0 always has this bit set to 0. This bit is read only from the PCI interface. This bit has no implementation in the S5920 other than providing it during a configuration read cycle.
2:1	Memory Type. These bits define whether the memory space is 32 or 64 bits wide and if the space location is restricted to be within the first megabyte of memory space. The encoding is as follows: **Bits 2 1** — **Description** 0 0 — Region is 32-bits wide and can be located anywhere in 32-bit memory space. 0 1 — Region is 32 bits wide and must be mapped below the first Mbytes of memory space. 1 0 — Region is 64 bits wide and can be mapped anywhere within 64-bit memory space. (Not supported by this device.) 1 1 — Reserved.
2	Note: The 64-bit memory space is not supported by this device. Bit 2 is hardwired to 0. Options are restricted to desired to memory space anywhere within 32-bit memory space or located in the first megabyte. For Base Addresses 1 through 4, this bit is cleared by the reset pin and later initialized by the external boot memory option.
0	Space Indicator = 0. When set to 0, this bit defines a base address region as memory space and the remaining bits in the base address register are defined as shown in Figure 11a.

Figure 11b. Base Address Register - I/O



Bit	Description
31:2	Base Address Location. These bits are used to position the decoded region in I/O space. Only bits which return a 1 after being written as 1 are usable for this purpose. Except for Base Address 0, these bits are individually enabled by the contents sourced from the external nvRAM.
1	Reserved. This bit should be 0. (Note: disabled Base Address Registers will return all 0s for the entire register location, bits 31 through 0).
0	Space Indicator = 1. When 1, this bit identifies a base address region as an I/O space and the remaining bits in the base address register have the definition as shown in Figure 11b.

Table 16. Base Address Register Response (Memory Assigned) to All-Ones Write Operation

Response	Size in Bytes	nvRAM boot value [1]
00000000h	none - disabled	00000000h or BIOS missing [2]
FFFFFFF0h	16 bytes (4 DWORDs)	FFFFFFF0h
FFFFFFE0h	32 bytes (8 DWORDs)	FFFFFFE0h
FFFFFFC0h	64 bytes (16 DWORDs)	FFFFFFC0h
FFFFFF80h	128 bytes (32 DWORDs)	FFFFFF80h
FFFFFF00h	256 bytes (64 DWORDs)	FFFFFF00h
FFFFFE00h	512 bytes (128 DWORDs)	FFFFFE00h
FFFFFC00h	1K bytes (256 DWORDs)	FFFFFC00h
FFFFF800h	2K bytes (512 DWORDs)	FFFFF800h
FFFFF000h	4K bytes (1K DWORDs)	FFFFF000h
FFFFE000h	8K bytes (2K DWORDs)	FFFFE000h
FFFFC000h	16K bytes (4K DWORDs)	FFFFC000h
FFFF8000h	32K bytes (8K DWORDs)	FFFF8000h
FFFF0000h	64K bytes (16K DWORDs)	FFFF0000h
FFFE0000h	128K bytes (32K DWORDs)	FFFE0000h
FFFC0000h	256K bytes (64K DWORDs)	FFFC0000h
FFF80000h	512K bytes (128K DWORDs)	FFF80000h
FFF00000h	1M bytes (256K DWORDs)	FFF00000h
FFE00000h	2M bytes (512K DWORDs)	FFE00000h
FFC00000h	4M bytes (1M DWORDs)	FFC00000h
FF800000h	8M bytes (2M DWORDs)	FF800000h
FF000000h	16M bytes (4M DWORDs)	FF000000h
FE000000h	32M bytes (8M DWORDs)	FE000000h
FC000000h	64M bytes (16M DWORDs)	FC000000h
F8000000h	128M bytes (32M DWORDs)	F8000000h
F0000000h	256M bytes (64M DWORDs)	F0000000h
E0000000h	512M bytes (128M DWORDs)	E0000000h

1. The two most significant bits define bus width for BADR1:4 in Pass-Thru operation. (See S5920 Base Address Register Definition.)
2. Bits D3, D2 and D1 may be set to indicate other attributes for the memory space.

Table 17. Read Response (I/O Assigned) to an All-Ones Write Operation to a Base Address Register

Response	Size in Bytes	nvRAM boot value
00000000h	none - disabled	00000000h or BIOS missing
FFFFFFFDh	4 bytes (1 DWORDs)	FFFFFFFDh
FFFFFFF9h	8 bytes (2 DWORDs)	FFFFFFF9h
FFFFFFF1h	16 bytes (4 DWORDs)	FFFFFFF1h
FFFFFFE1h	32 bytes (8 DWORDs)	FFFFFFE1h
FFFFFFC1h	64 bytes (16 DWORDs)	FFFFFFC1h
FFFFFF81h	128 bytes (32 DWORDs)	FFFFFF81h [1]
FFFFFF01h	256 bytes (64 DWORDs)	FFFFFF01h

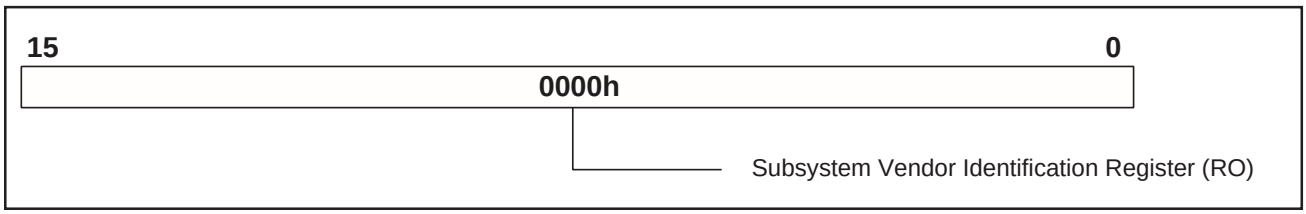
1. Base Address Register 0, at offset 10h, powers up as FFFFFF81h. This default assignment allows usage without an external boot memory. Should an nvRAM be used, the base address can be boot loaded to become a memory space (FFFFFF80h or FFFFFF82h).

SUBSYSTEM VENDOR IDENTIFICATION REGISTER (SVID)

Register Name: Subsystem Vendor ID
Address Offset: 2Ch-2Dh
Power-up value: 0000h
Boot-load: External nvRAM offset 6Ch-6Dh
Attribute: Read Only (RO)
Size: 16 bits

This register is used to uniquely identify the user board or subsystem. It provides a mechanism for add-in card vendors to distinguish devices with the same Vendor ID and Device ID. Implementation of this register is mandatory for 2.2 compliance and an all-zero value indicates that the device does not support subsystem identification. Subsystem Vendor IDs may be obtained directly from the PCI SIG by the user and it is loaded by the S5920 from the external nvRAM at power up.

Figure 12. Subsystem Vendor Identification Register



Bit	Description
15:0	Subsystem Vendor Identification Number.

SUBSYSTEM ID REGISTER (SID)

Register Name: Subsystem Identification

Address Offset: 2Eh-2Fh

Power-up value: 0000h

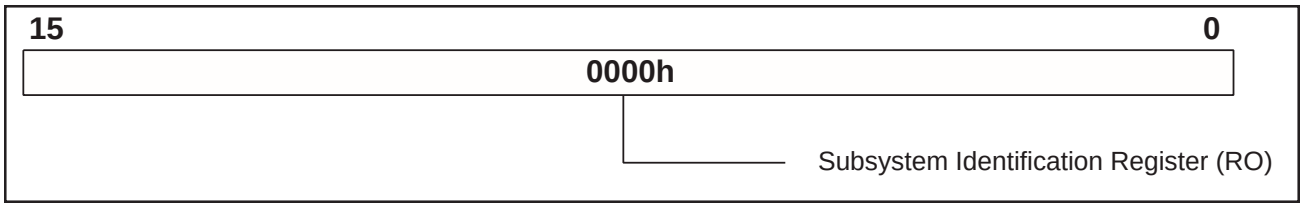
Boot-load: External nvRAM offset 6Eh-6Fh

Attribute: Read Only (RO)

Size: 16 bit

This register is used to further identify the add-in board or subsystem. It provides a mechanism for add-in card vendors to distinguish devices with the same Vendor ID and Device ID. Implementation of this register is mandatory for 2.2 compliance and an all-zero value indicates that the device does not support subsystem identification. Subsystem ID is vendor-specific. It is loaded by the S5920 from the external nvRAM at power up.

Figure 13. Subsystem Identification Register



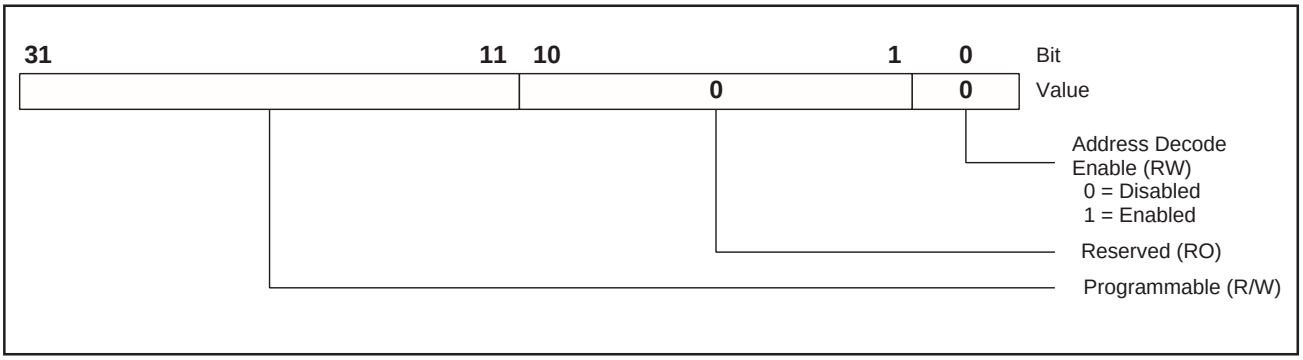
Bit	Description
15:0	Subsystem Identification Number.

EXPANSION ROM BASE ADDRESS REGISTER (XROM)

Register Name:	Expansion ROM Base Address
Address Offset:	30h
Power-up value:	00000000h
Boot-load:	External nvRAM offset 70h
Attribute:	bits 31:11, bit 0 Read/Write; bits 10:1 Read Only
Size:	32 bits

The Expansion ROM Base Address Register provides a mechanism for assigning a space within physical memory for a BIOS expansion ROM. Access from the PCI bus to the memory space defined by this register will cause one or more accesses to the S5920 external nvRAM interface. Since PCI bus accesses to the ROM may be 32 bits wide, repeated read operations to the ROM are generated, and the wider data is assembled internal to the S5920 controller. The data is then transferred to the PCI bus by the S5920. Only memory read cycles should be performed to this location.

Figure 14. Expansion ROM Base Address Register



Bit	Description
31:11	Expansion ROM Base Address Location. These bits are used to position the decoded region in memory space. Only bits which return a 1 after being written as 1 are usable for this purpose. These bits are individually enabled by the contents sourced from the external boot memory (nvRAM). The desired size for the ROM memory is determined by writing all ones to this register and then reading back the contents. The number of bits returned as zeros, in order from least significant to most significant bit, indicates the size of the expansion ROM. This controller limits the expansion ROM area to 2K bytes (due to the serial nvRAM's limit of 11 bits of address). The allowable returned values after all ones are written to this register are shown in Table 18.
10:1	Reserved. All zeros.
0	Address Decode Enable. The Expansion ROM address decoder is enabled or disabled with this bit. When this bit is set, the decoder is enabled. When this bit is cleared, the decoder is disabled. It is requiredt the PCI command register (PCICMD) also have the memory decode bit enabled for this bit to have any effect. In addition, the corresponding bit must be set in the external nvRAM (see page 2-74, Table 1). If not set, the PCI host cannot enable/disable this Address Decode bit.

Table 18. Read Response to Expansion ROM Base Address Register (after all ones written)

Response	Size in Bytes	nvRAM boot value
00000000h	none - disabled	00000000h or BIOS missing [1]
FFFFF801h	2K bytes (512 DWORDs)	FFFFF801h

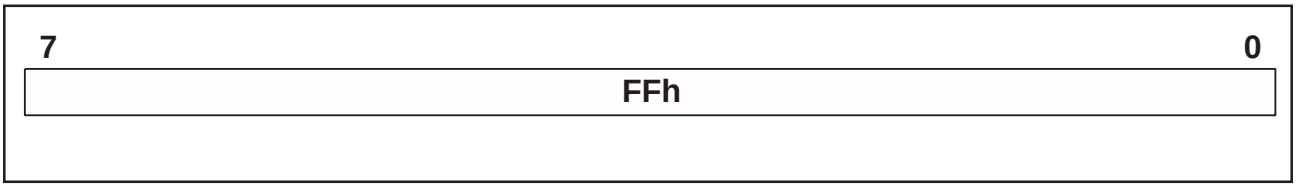
1. The Expansion ROM Base Address Register nvRAM boot value is internally hardwired to FFFFF80Xh, where X = 000xb (i.e., only the least-significant bit, or Address Decode Enable bit, is programmable). This defines both the minimum and maximum expansion ROM size supported by the S5920 (2K bytes). The Address Decode Enable bit in the nvRAM (the LSB) must be set to enable this region. If not set, a PCI Configuration read of this region will always respond with 00000000h.

INTERRUPT LINE REGISTER (INTLN)

Register Name:	Interrupt Line
Address Offset:	3Ch
Power-up value:	FFh
Boot-load:	External nvRAM offset 7Ch
Attribute:	Read/Write
Size:	8 bits

This register indicates the interrupt routing for the S5920 controller. The ultimate value for this register is system-architecture specific. For x86 based PCs, the values in this register correspond with the established interrupt numbers associated with the dual 8259 controllers used in those machines. In x86-based PC systems, the values of 0 to 15 correspond with the IRQ numbers 0 through 15, and the values from 16 to 254 are reserved. The value of 255 (the controller's default power-up value) signifies either "unknown" or "no connection" for the system interrupt. This register is boot-loaded from the external boot memory or may be written by the PCI interface.

Figure 15. Interrupt Line Register

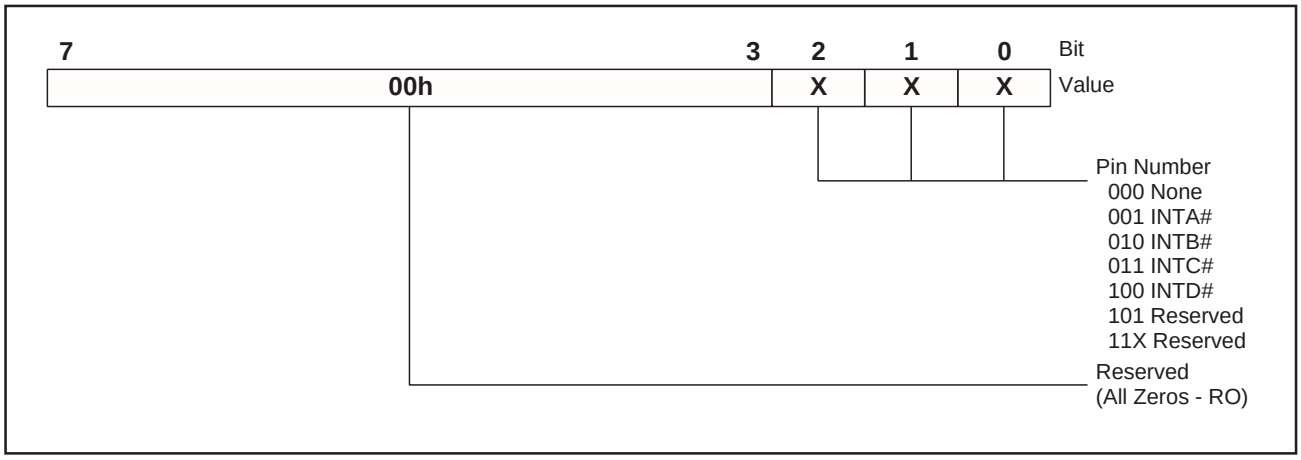


INTERRUPT PIN REGISTER (INTPIN)

Register Name	Interrupt Pin
Address Offset:	3Dh
Power-up value:	01h
Boot-load:	External nvRAM offset 7Dh
Attribute:	Read Only
Size:	8 bits

This register identifies which PCI interrupt, if any, is connected to the controller's PCI interrupt pins. The allowable values are 0 (no interrupts), 1 (INTA#), 2 (INTB#), 3 (INTC#), and 4 (INTD#). The default power-up value assumes INTA#.

Figure 16. Interrupt Pin Register



MINIMUM GRANT REGISTER (MINGNT)

Register Name: Minimum Grant

Address Offset: 3Eh

Power-up value: 00h, hardwired

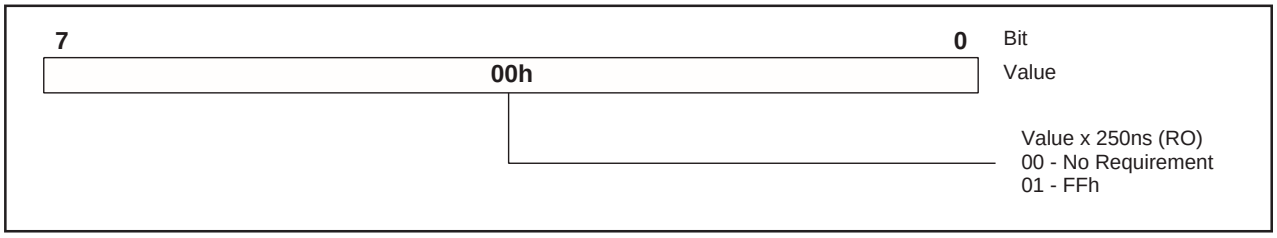
Boot-load: not used

Attribute: Read Only

Size: 8 bits

This register may be optionally used by bus masters to specify how long a burst period the device needs. A value of zero indicates that the bus master has no stringent requirement. The units defined by the least significant bit are in 250 ns increments. This register is treated as “information only” since the S5920 is a PCI target device only.

Figure 17. Minimum Grant Register



MAXIMUM LATENCY REGISTER (MAXLAT)

Register Name: Maximum Latency

Address Offset: 3Fh

Power-up value: 00h, hardwired

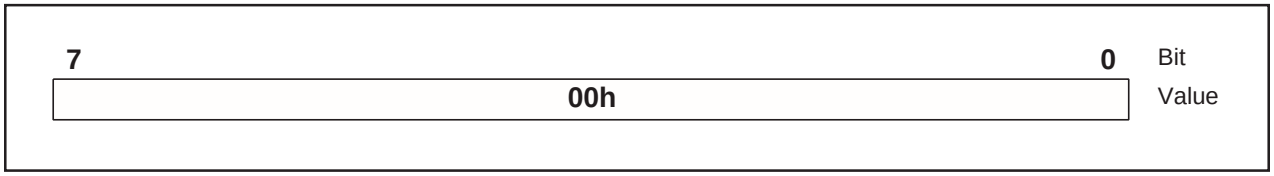
Boot-load: not used

Attribute: Read Only

Size: 8 bits

This register may be optionally used by bus masters to specify how often this device needs PCI bus access. A value of zero indicates that the bus master has no stringent requirement. The units defined by the least significant bit are in 250 ns increments. Since the S5920 is a PCI target device only, this register is treated as “information only” and has no further implementation within this device.

Figure 18. Maximum Latency Register



OPERATION REGISTERS

All S5920 control and communications are performed through two register groups: PCI Operation Registers Add-On Operation Registers. Some registers in both groups are accessible from both buses. This chapter describes the PCI Operation Register set first and then the Add-On Operation Register set for easier understanding. An access to a register common to both buses at the same time is not allowed. Unpredictable behavior may occur.

PCI BUS OPERATION REGISTERS

The PCI bus operation registers are mapped as 6 DWORD registers located at the address space (I/O or memory) specified by Base Address Register 0. These locations are the primary method of communication between the PCI and Add-On buses. It is NOT recommended to read or write from an undefined address. The read results and write effects cannot be guaranteed. Table 1 lists the PCI Bus Operation Registers.

Table 1. Operation Registers - PCI Bus

Address Offset	Abbreviation	Register Name
0Ch	OMB	Outgoing Mailbox Register
1Ch	IMB	Incoming Mailbox Register
34h	MBEF	Mailbox Empty/Full Status Register
38h	INTCSR	Interrupt Control/Status Register
3Ch	RCR	Reset Control Register
60h	PTCR	Pass-Thru Configuration Register

Note: Absolute register address locations are acquired by adding BADR0 to the “address offset” listed above.

OUTGOING MAILBOX REGISTER (OMB)

Register Names: Outgoing Mailbox

PCI Address Offset: 0Ch

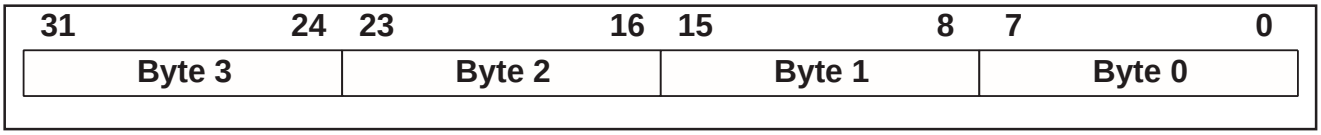
Power-up value: Undefined

PCI Attribute: Read/Write

Size: 32 bits

This DWORD register provides a method for sending command or parameter data to the Add-On bus. PCI bus transactions to this register may be of any width (byte, word, or DWORD). Writing to this register can be a source for Add-On bus interrupts by enabling interrupt generation through the use of the Add-on's Interrupt Control/Status Register. This is also called the Add-On Incoming Mailbox Register (AIMB). Reading from this register will not affect interrupts or the MBEF status register.

Figure 1. Outgoing Mailbox



PCI INCOMING MAILBOX REGISTER (IMB)

Register Names: Incoming Mailbox

PCI Address Offset: 1Ch

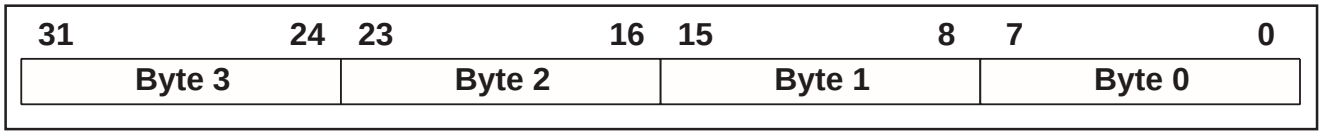
Power-up value: Undefined

PCI Attribute: Read Only

Size: 32 bits

This DWORD register provides a method for receiving user-defined status or parameter data from the Add-On bus. PCI bus transactions to this register may be of any width (byte, word, or DWORD). Only read operations are supported from this register. Reading from this register can be a source for Add-On bus interrupt by enabling interrupt generation through the use of the Add-On Interrupt Control/Status Register. Byte 3 of this mailbox can also be controlled via external hardware from the Add-On bus. This register is also referred to as the Add-On Outgoing Mailbox Register AOMB).

Figure 2. Incoming Mailbox



PCI MAILBOX EMPTY/FULL STATUS REGISTER (MBEF)

Register Name: Mailbox Empty/Full Status

PCI Address Offset: 34h

Power-up value: 00000000h

PCI Attribute: Read Only

Size: 32 bits

This register provides empty/full visibility for each byte within the mailboxes. The empty/full status for the PCI Outgoing mailbox is displayed on bits 15 to 12 and the empty/full status for the PCI Incoming mailbox is presented on bits 31 to 28. A value of 1 signifies that a given mailbox has been written by one bus interface but has not yet been read by the corresponding destination interface. The PCI bus incoming mailbox transfers data from the Add-On bus to the PCI bus, and the PCI outgoing mailbox transfers data from the PCI bus to the Add-On bus. This register is also referred to as the Add-On Mailbox Empty/Full Status Register (AMBEF).

Figure 3. Mailbox Empty/Full Status Register (MBEF)

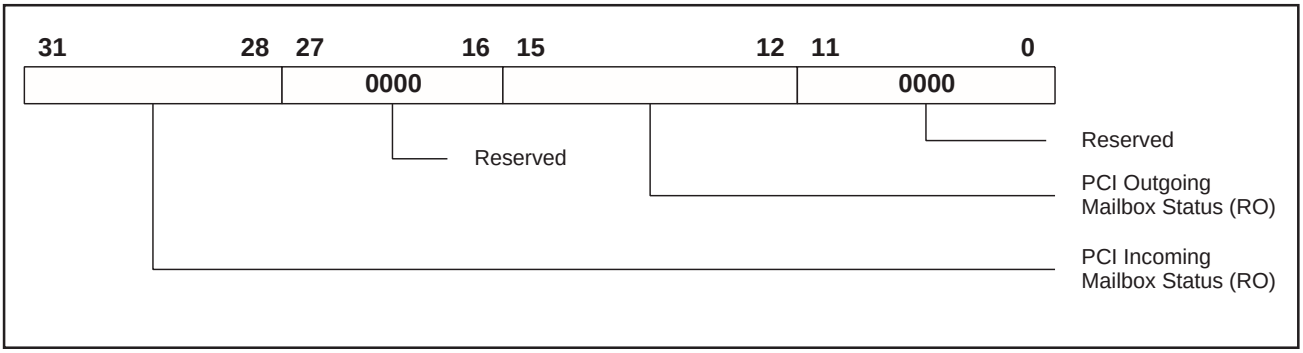


Table 2. Mailbox Empty/Full Status Register

Bit	Description
31:28	PCI Incoming Mailbox Status. This field indicates which byte of the incoming mailbox register has been written by the Add-On interface but has not been read by the PCI bus. Each bit location corresponds to a specific byte within the incoming mailbox. A value of one for each bit signifies that the specified mailbox byte is full, and a value of 0 signifies empty. The mapping of these status bits to bytes within the mailbox is as follows: Bit 31 = Incoming mailbox byte 3 Bit 30 = Incoming mailbox byte 2 Bit 29 = Incoming mailbox byte 1 Bit 28 = Incoming mailbox byte 0
15:12	PCI Outgoing Mailbox Status. This field indicates which byte of the outgoing mailbox register has been written by the PCI bus interface but has not yet been read by the Add-On bus. Each bit location corresponds to a specific byte within the outgoing mailbox. A value of one for each bit signifies that the specified mailbox byte is full, and a value of 0 signifies empty. The mapping of these status bits to bytes is as follows: Bit 15 = Outgoing mailbox byte 3 Bit 14 = Outgoing mailbox byte 2 Bit 13 = Outgoing mailbox byte 1 Bit 12 = Outgoing mailbox byte 0

PCI INTERRUPT CONTROL/STATUS REGISTER (INTCSR)

Register Name: Interrupt Control and Status

PCI Address Offset: 38h

Power-up value: 00000C0Ch

PCI Attribute: Read/Write, Read/Write Clear

Size: 32 bits

This register configures the conditions which will produce an interrupt on the PCI bus interface, a method for viewing the cause of the interrupt, and a method for acknowledging (removing) the interrupt's assertion.

Interrupt sources:

- The Outgoing mailbox becomes empty.
- The Incoming mailbox becomes full.
- Add-On interrupt pin enable and flag.

Figure 4. Interrupt Control Status Register

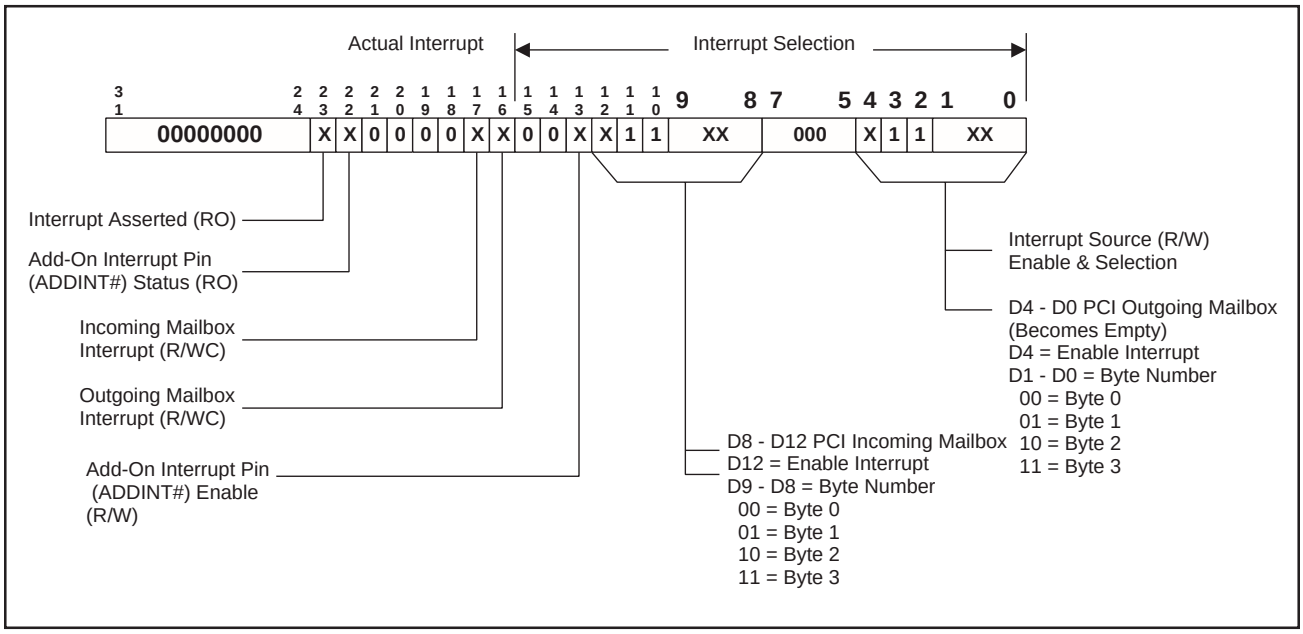


Table 3. Interrupt Control Status Register

Bit	Description
31:24	Reserved. Always zero.
23	Interrupt Asserted. This read only status bit indicates that one or more of the three possible interrupt conditions are present. This bit is the OR of the mailbox interrupt conditions described by Bits 17 and 16, as well as the OR of the Add-On interrupt described in Bit 22 (if the Add-On Interrupt is Enabled with Bit 13). No PCI interrupt is generated, nor is this bit ever set, for an Add-On Interrupt without the Add-On Interrupt Enable set.
22	Add-On Interrupt. This bit is set when the ADDINT# input pin is driven low by an Add-On bus device. A high bit indicates an Add-On device is requesting service. In addition, if the ADDINT# Enable bit is set, the S5920 will assert a PCI interrupt (INTA# driven low). The source driving ADDINT# must deassert this input before the PCI interrupt (INTA#) is driven to a false state. Host software must clear the Add-On interrupt source before exiting its interrupt handler routine.
21:18	Reserved. Always zero.
17	PCI Incoming Mailbox Interrupt. This bit can be set when the mailbox is written by the Add-On interface. This bit operates as read or write one clear. A write to this bit with the data of “one” will cause this bit to be reset; a write to this bit with the data of “0 ” will not change the state of this bit.
16	PCI Outgoing Mailbox Interrupt. This bit can be set when the mailbox is read by the Add-On interface. This bit operates as read or write one clear. A write to this bit with the data of “one” will cause this bit to be reset; a write to this bit with the data of “0 ” will not change the state of this bit.
15:14	Reserved. Always zero.
13	ADDINT# Enable. If this bit is high, the S5920 will allow the Add-On interrupt request to drive the INTA# pin. It has no effect on the assertion of the Add-On Interrupt Bit 22.
12	Enable Incoming Mailbox interrupt. This bit allows a write from the incoming mailbox register byte identified by bits 9 and 8 to produce a PCI interface interrupt. This bit is read/write.
11:10	Hardwired to 11. Reserved.
9:8	Incoming Mailbox Byte Interrupt Select. This field selects which byte of the mailbox is to actually cause the interrupt. [00]b selects byte 0, [01]b selects byte 1, [10]b selects byte 2, and [11]b selects byte 3. This field is read/write.
7:5	Reserved. Always zero.
4	Enable Outgoing Mailbox Interrupt. This bit allows a read by the Add-On of the outgoing mailbox register byte identified by bits 1 and 0 to produce a PCI interface interrupt. This bit is read/write.
3:2	Hardwired to 1. Reserved
1:0	Outgoing Mailbox Byte Interrupt Select. This field selects which byte of the mailbox is to actually cause the interrupt. [00]b selects byte 0, [01]b selects byte 1, [10]b selects byte 2, and [11]b selects byte 3. This field is read/write.

PCI RESET CONTROL REGISTER (RCR)

Register Name: Reset Control Register

PCI Address Offset: 3Ch

Power-up value: 00000000h

Attribute: Read/Write, Read Only, Write Only

Size: 32 bits

This register provides a method to perform software resets. It will also control nvRAM accesses.

The following controls are available:

- Assert reset to Add-On
- Reset mailbox empty full status flags
- Write/Read external non-volatile memory
- Reset Pass-Thru Read FIFO

Figure 5. FIFO Control/Status Register

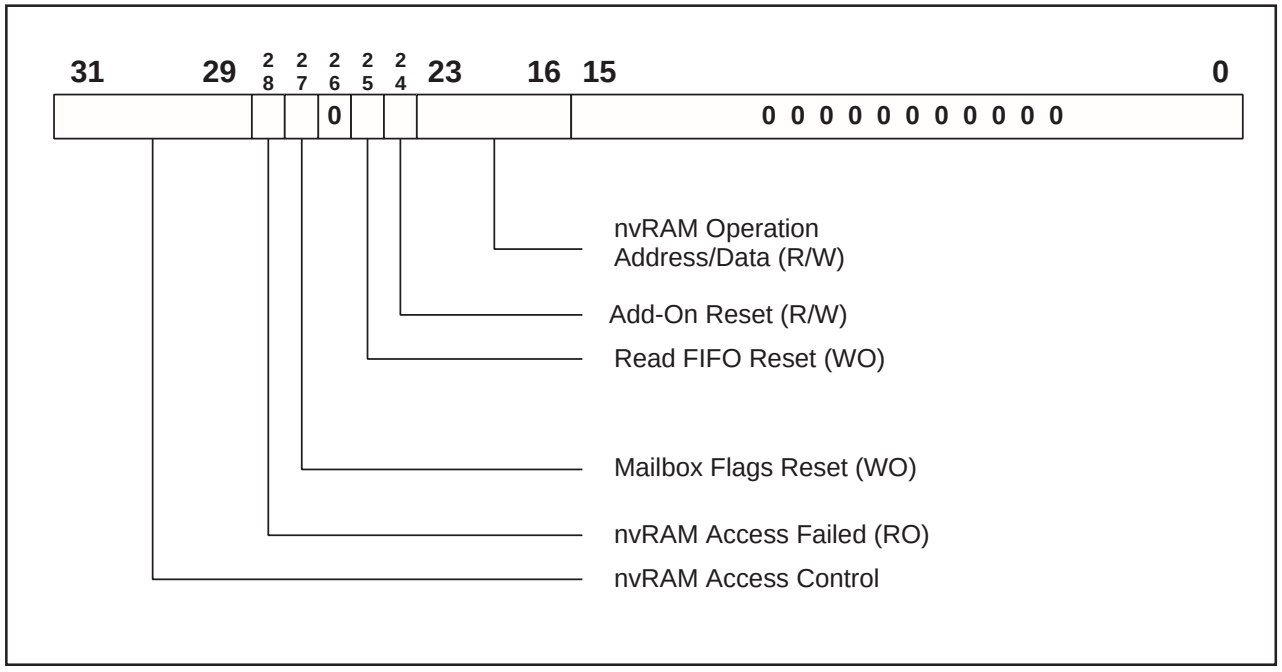


Table 4. Reset Control Register

Bit	Description
31:29	nvRAM Access Control. This field provides a method for access to the optional external non-volatile memory. Write operations are achieved by a sequence of byte operations involving these bits and the 8-bit field of bits 23 through 16. The sequence requires that the low-order address, high-order address, and then a data byte are loaded in order. Bit 31 of this field acts as a combined enable and ready for the access to the external memory. D31 must be written to a 1 before an access can begin, and subsequent accesses must wait for bit D31 to become 0 (ready). D31 D30 D29 W/R 0 X X W Inactive 1 0 0 W Load low address byte 1 0 1 W Load high sddress byte 1 1 0 W Begin write 1 1 1 W Begin read 0 X X R Ready 1 X X R Busy Cautionary note: The nonvolatile memory interface is also available for access by the Add-On interface. While simultaneous accesses to the nv memory by both the Add-On and PCI are supported (via arbitration logic), software must be designed to prevent the possibility of data corruption within the memory and to provide for accurate data retrieval.
28	nvRAM Access Failed. Indicate the last nvRAM access failed. This flag is cleared automatically upon the start of the next read/write operation.
27	Mailbox Flag Reset. Writing a one to this bit causes all mailbox status flags to become reset (EMPTY). It is not necessary to write this bit to 0 afterwards because it is used internally to produce a reset pulse. Since reading this bit will always return a 0, this bit is write only.
26	Reserved. Always zero.
25	Read FIFO Reset. Writing a one to this bit causes the read FIFO to reset (empty). It is not necessary to write a 0 to this bit. This bit is write only. This feature is intended for test only. However, it can be used during operation if several PCI idle cycles are inserted following the assertion of this command.
24	Add-On Pin Reset. Writing a one to this bit causes the reset output pin to become active (SYSRST#). Clearing this bit is necessary in order to remove the assertion of reset. This bit is read/write.
23:16	Non-volatile Memory Address/Data Port. This 8-bit field is used in conjunction with bits 31, 30 and 29 of this register to access the external non-volatile memory. The contents written are either low address, high address, or data as defined by bits 30 and 29. This register will contain the external non-volatile memory data when the proper read sequence for bits 31 through 29 is performed.
15:0	Reserved. Always zero.

PCI PASS-THRU CONFIGURATION REGISTER (PTCR)

Register Name: Pass-Thru Configuration Register

PCI Address Offset: 60h

Power-up value: 80808080h

PCI Attribute: Read/Write

Size: 32 bits

This register controls the configuration for Pass-Thru Regions 1-4:

Byte 0 Controls Pass-Thru Region 1
Byte 1 Controls Pass-Thru Region 2
Byte 2 Controls Pass-Thru Region 3
Byte 3 Controls Pass-Thru Region 4

IMPORTANT NOTE: This register (PTCR) is physically the same as the Add-On Pass-Thru Configuration Register (APTCR). It is intended that either the PCI system or local Add-On interface will write to this register, but not both. However, in the event that both the PCI and Add-On must write to this register, whichever side wrote last will update its value.

Also, Pass-Thru operation cannot be guaranteed if this register is updated while a Pass-Thru operation is already in progress.

Figure 6. Pass-Thru Configuration Register

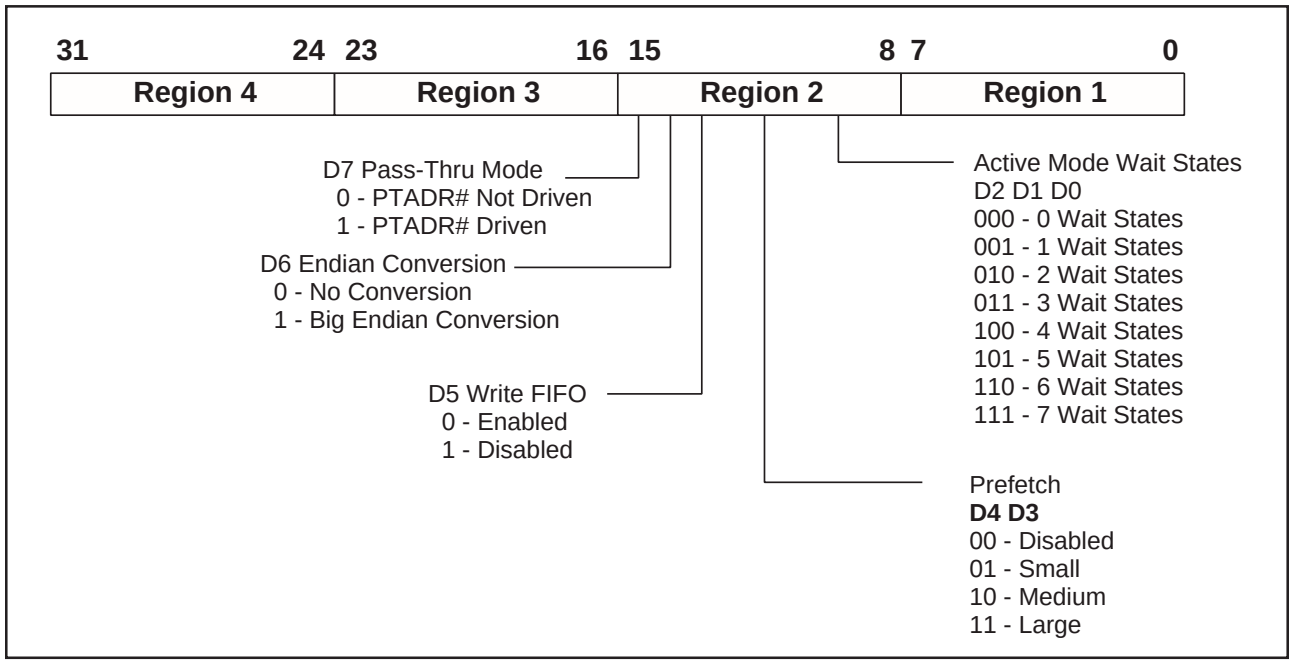


Table 5 describes one of the four configuration registers. All four region configuration registers are exactly the same.

Table 5. Pass-Thru Configuration Register

Bit	Description
7	PTADR# mode. This bit is only valid in Active mode. If this bit is 0 , PTADR# is not driven at the beginning of an active cycle. If this bit is set to 1 (default state), the S5920 will assert PTADR# for one clock cycle after PTATN# is asserted. The Pass-Thru address is also driven while PTADR# is low. This bit is a don't care if the device is operating in Passive mode.
6	Endian conversion. If this bit is set to one, the S5920 will convert the Add-On bus from the default little endian format to a big endian format. Reference Chapter 9 for more details.
5	Write FIFO disabled. If this bit is set to 1, the S5920 will not accept the next piece of data (on a PCI write) until the Add-On has accepted the previous piece of data. If this bit is set to 0, the S5920 will accept data from the PCI until the Pass-Thru write FIFO is full.
4:3	Prefetch. These bits control the number of DWORDs the S5920 will prefetch after the current PCI Pass-Thru read completes. The actual amount of data prefetched depends upon any number of different scenarios. The prefetch values of "small", "medium" and "large" are available to tune the system to achieve best overall performance (i.e., optimize PCI bus transfers or optimize Add-On bus transfers). The Pass-Thru read FIFO can be enabled to prefetch in either Active mode or Passive mode.
2:0	Wait states. In Active mode, the user can program the number of wait states required by the Add-On bus to complete a transaction. Up to 7 wait states can be programmed (per region). The S5920 will count the number of clocks programmed into this register before finishing the current data transaction if PTWAIT# is high. If PTWAIT# is driven low, additional wait states may be inserted. Bits 2, 1 and 0 are don't care if operating in Passive mode.

ADD-ON BUS OPERATION REGISTERS

The Add-On bus interface provides access to 8 DWORDs of data, control and status information. All of these locations are accessed by asserting the Add-On bus chip select pin (SELECT#) and the byte-enable pins (BE[3:0]), in conjunction with either the read or write control enables (signal pin RD# or WR#). All registers are accessed with signals synchronous to the Add-On clock.

This register group represents the primary method for communication between the Add-On and PCI buses as viewed by the Add-On. The flexibility of this arrangement allows a number of user-defined software protocols to be built. One should NOT read/write from any undefined address, or the read results and write effects cannot be guaranteed. Table 6 lists the Add-On Bus Operation Registers.

Table 6. Operation Registers - Add-On Interface

Address Offset	Abbreviation	Register Name
0Ch	AIMB	Add-On Incoming Mailbox Register
1Ch	AOMB	Add-On Outgoing Mailbox Register
28h	APTA	Add-On Pass-Thru Address Register
2Ch	APTD	Add-On Pass-Thru Data Register
34h	AMBEF	Add-On Mailbox Empty/Full Status Register
38h	AINT	Add-On Interrupt Control/Status Register
3Ch	ARCR	Add-On Reset Control Register
60h	APTCR	Add-On Pass-Thru Configuration Register

ADD-ON INCOMING MAILBOX REGISTER (AIMB)

Register Names: Incoming Mailbox
Add-On Address: 0Ch
Power-up value: XXXXXXXXh
Add-On Attribute: Read Only
Size: 32 bits

This DWORD register provides a method for receiving user-defined status or parameter data from the PCI system. Add-On bus read operations to this register may be of any width (byte, word, or DWORD). Only read operations are supported. Reading from this register can optionally cause a PCI bus interrupt (if desired) by enabling interrupt generation through the use of the PCI's Interrupt Control/Status Register. This register is also referred to as the PCI Outgoing Mailbox Register.

ADD-ON OUTGOING MAILBOX REGISTER (AOMB)

Register Names: Outgoing Mailbox
Add-On Address: 1Ch
Power-up value: XXXXXXXXh
Add-On Attribute: Read/Write
Size: 32 bits

This DWORD register provides a method for sending command or parameter data to the PCI interface. Add-On bus operations to this register may be of any width (byte, word, or DWORD). Writing to this register can be a source for PCI bus interrupts (if desired) by enabling interrupt generation through the use of the PCI's Interrupt Control/Status Register. This is also called the PCI Incoming Mailbox Register (IMB). Byte 3 of this mailbox can also be controlled via the external mailbox port. Reading from this register will not affect interrupts or the AMBEF Status Register. (OMB).

ADD-ON PASS-THRU ADDRESS REGISTER (APTA)

Register Name: Add-On Pass-Thru Address
Add-On Address: 28h
Power-up value: XXXXXXXXh
Add-On Attribute: Read Only
Size: 32 bits

This register stores the address of any active Pass-Thru PCI bus cycle that has been accepted by the S5920. When one of the base address decode registers 1-4 encounters a PCI bus cycle which selects the region defined by it, this register stores that current cycle's active address. This address is incremented after every 32-bit Pass-Thru data transfer.

ADD-ON PASS-THRU DATA REGISTER (APTD)

Register Name: Add-On Pass-Thru Data
Add-On Address: 2Ch
Power-up value: XXXXXXXXh
Add-On Attribute: Read/Write
Size: 32 bits

This register, along with APTA register, is used to perform Pass-Thru transfers. When one of the base address decode registers 1-4 encounters a PCI bus cycle which selects the region defined by it, the APTA register will contain that current cycle's active address and the APTD will contain the data (PCI bus writes) or must be written with data (PCI bus reads). Wait states are generated on the PCI bus until this register is read (PCI bus writes) or this register is written (PCI bus reads) when in Passive mode.

ADD-ON MAILBOX EMPTY/FULL STATUS REGISTER (AMBEF)

Register Name: Mailbox Empty/Full Status

Add-On Address: 34h

Power-up value: 00000000h

Add-On Attribute: Read Only

Size: 32 bits

This register provides empty/full visibility for each byte within the mailboxes. The empty/full status for the Add-On Incoming mailbox is displayed on bits 15 to 12 and the empty/full status for the Add-On Outgoing mailbox is presented on bits 31 to 28. A value of 1 signifies that a given mailbox has been written by one bus interface but has not yet been read by the corresponding destination interface. The Add-On bus incoming mailbox is used to transfer data from the PCI bus to the Add-On bus, and the Add-On outgoing mailbox is used to transfer data from the Add-On bus to the PCI bus. This register is also referred to as the PCI Mailbox Empty/Full Status Register (MBEF).

Figure 7. Mailbox Empty/Full Status Register

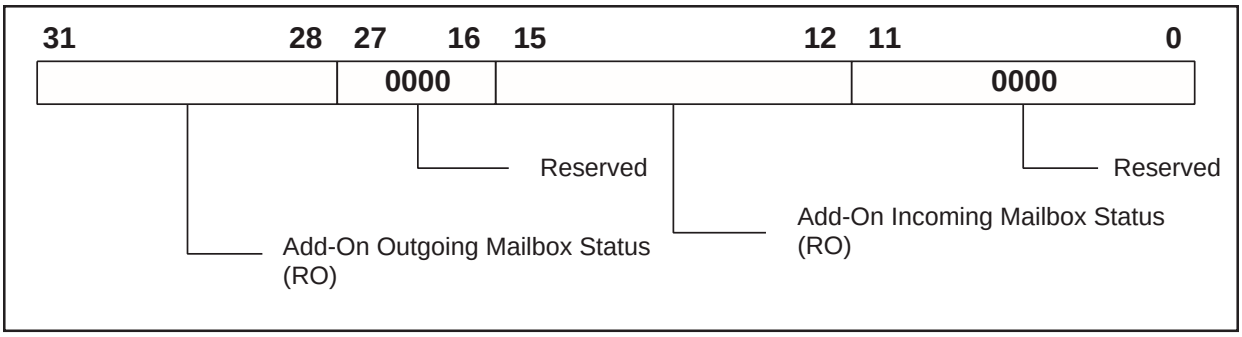


Table 7. Mailbox Empty/Full Status Register

Bit	Description
31:28	Add-On Outgoing Mailbox Status. This field indicates which byte of the outgoing mailbox register has been written by the Add-On interface but has not yet been read by the PCI bus. Each bit location corresponds to a specific byte within the outgoing mailbox. A value of 1 for each bit signifies that the specified mailbox byte is full, and a value of 0 signifies empty. The mapping of these status bits to bytes within the mailbox is as follows: Bit 31 = Outgoing mailbox byte 3 Bit 30 = Outgoing mailbox byte 2 Bit 29 = Outgoing mailbox byte 1 Bit 28 = Outgoing mailbox byte 0
15:12	Add-On Incoming Mailbox Status. This field indicates which byte of the incoming mailbox register has been written by the PCI bus interface but has not yet been read by the Add-On bus. Each bit location corresponds to a specific byte within the incoming mailbox. A value of 1 for each bit signifies that the specified mailbox byte is full, and a value of 0 signifies empty. The mapping of these status bits to bytes is as follows: Bit 15 = Incoming mailbox byte 3 Bit 14 = Incoming mailbox byte 2 Bit 13 = Incoming mailbox byte 1 Bit 12 = Incoming mailbox byte 0

ADD-ON INTERRUPT CONTROL/STATUS REGISTER (AINT)

Register Name: Add-On Interrupt Control and Status

Add-On Address: 38h

Power-up value: 00000C0Ch

Attribute: Read/Write, Read/Write Clear

Size: 32 bits

This register provides the method for choosing which conditions are to produce an interrupt on the Add-On bus interface, a method for viewing the cause for the interrupt, and a method for acknowledging (removing) the interrupt's assertion.

Interrupt sources:

- Incoming mailbox becomes full
- Outgoing mailbox becomes empty
- Built-in self test issued

Figure 8. Add-On Interrupt Control Status Register

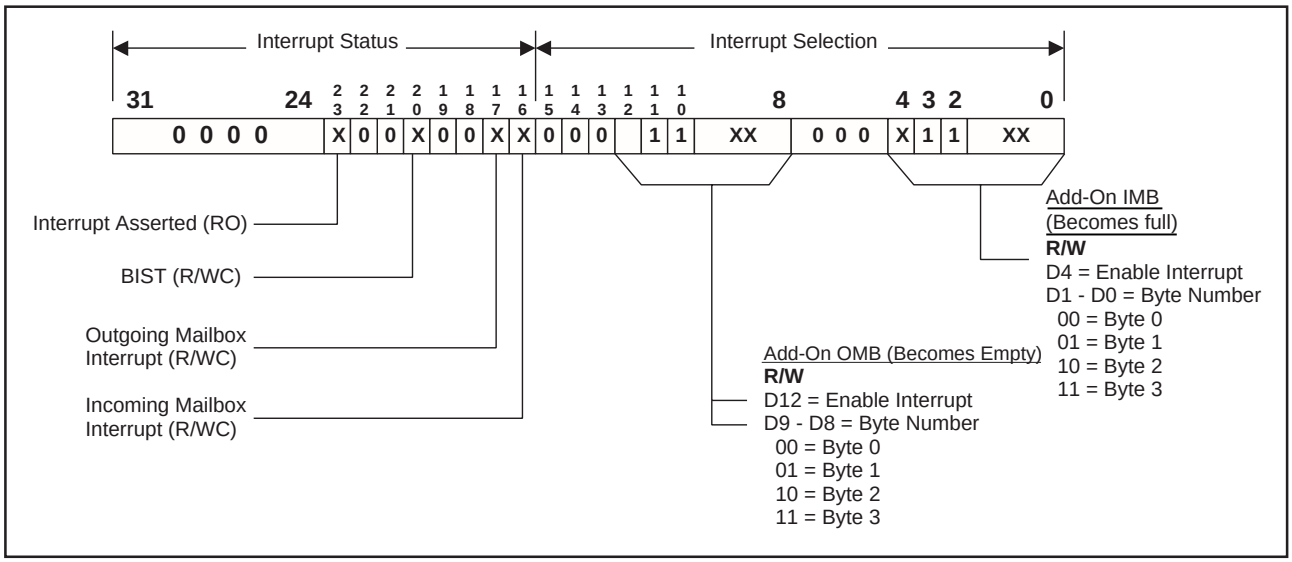


Table 8. Interrupt Control Status Register

Bit	Description
31:24	Reserved. Always zero.
23	Interrupt Asserted. This read-only status bit indicates that one or more interrupt conditions are present. This bit is the OR of the interrupt sources described by bits 20, 17 and 16 of this register.
22:21	Reserved. Always zero.
20	BIST. Built-In Self-Test Interrupt. This interrupt occurs when a self test is initiated by the PCI interface by writing to the PCI configuration register BIST. This bit will stay set until cleared by writing a 1 to this location. Self test completion codes may be passed to the PCI BIST register by writing to the ARCR register.
19:18	Reserved. Always zero.
17	Outgoing Mailbox Interrupt. This bit can be set when the mailbox is read by the PCI interface. This bit operates as read or write 1 clear. A write with the data as 1 will cause this bit to be reset; a write with the data as 0 will not change the state of this bit.
16	Incoming Mailbox Interrupt. This bit can be set when the mailbox is written by the PCI interface. This bit operates as read or write 1 clear. A write with the data as 1 will cause this bit to be reset; a write with the data as 0 will not change the state of this bit.
15:13	Reserved. Always zero.
12	Enable Outgoing Mailbox Interrupt. This bit allows a PCI read of the outgoing mailbox register to produce an Add-On interrupt. This bit is read/write.
11:10	Hardwired to 11.
9:8	Outgoing Mailbox Byte Interrupt Select. This field selects which byte of the mailbox is to cause the interrupt. [00]b selects byte 0, [01]b selects byte 1, [10]b selects byte 2, and [11]b selects byte 3. This field is read/write.
7:5	Reserved. Always zero.
4	Enable Incoming Mailbox Interrupt. This bit allows a write from the PCI bus to the incoming mailbox register to produce an Add-On interrupt. This bit is read/write.
3:2	Hardwired to 1.
1:0	Incoming Mailbox Byte Interrupt Select. This field selects which byte of the mailbox is to cause the interrupt. 00b selects byte 0, 01b selects byte 2, and 11b selects byte 3. This field is read/write.

ADD-ON RESET CONTROL REGISTER (ARCR)

Register Name:	Add-On Reset Control and Status
Add-On Address:	3Ch
Power-up value:	00h
Attribute:	Read/Write, Read Only, Write Only
Size:	32 bits

This register provides a method to perform software resets and nvRAM accesses.

The following Add-On controls are provided:

- Reset mailbox empty full status flags
- Reset Pass-Thru read FIFO
- Read/Write external non-volatile memory

Figure 9. Add-On General Control/Status Register

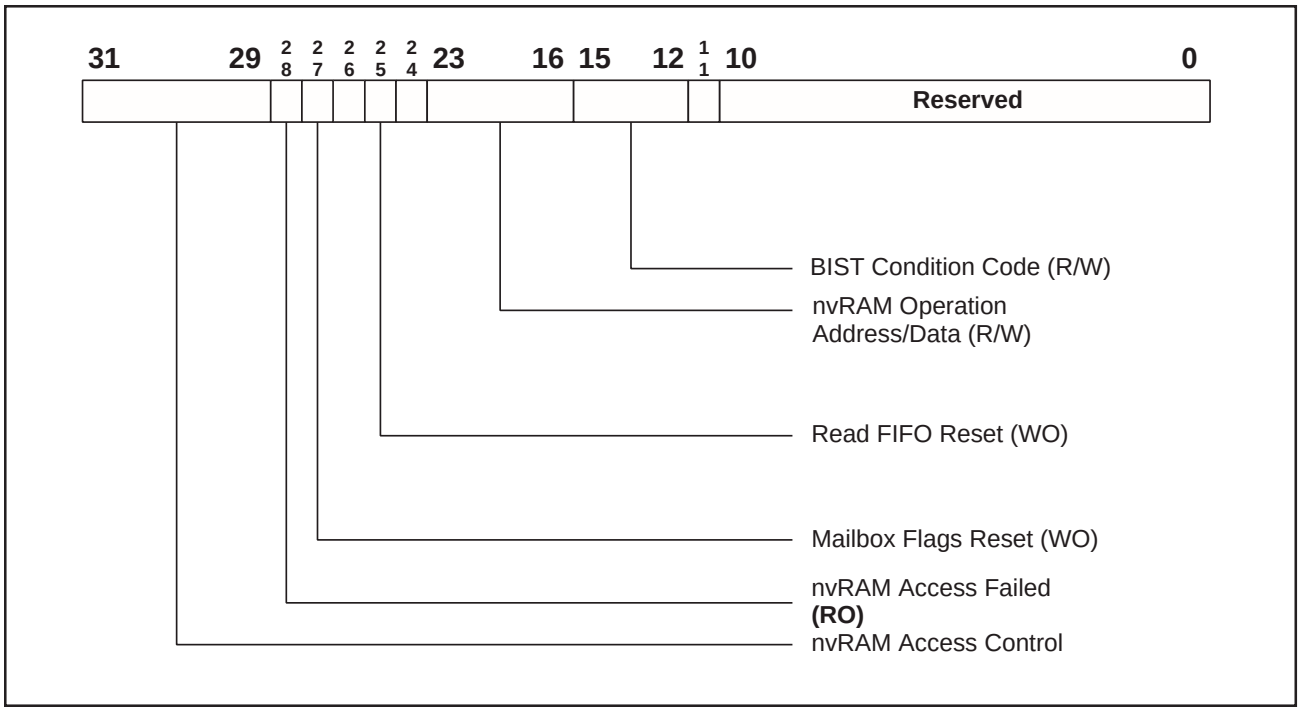


Table 9. Reset General Control/Status Register

Bit	Description
31:29	nvRAM Access Control. This field provides a method for access to the optional external non-volatile memory. Write operations are achieved by a sequence of byte operations involving these bits and the 8-bit field of bits 23 through 16. The sequence requires that the low-order address, high-order address, and then a data byte are loaded in order. Bit 31 of this field acts as a combined enable and ready for the access to the external memory. D31 must be set to a 1 before an access can begin, and subsequent accesses must wait for bit D31 to become 0 (ready). D31 D30 D29 W/R 0 X X W Inactive 1 0 0 W Load low address byte 1 0 1 W Load high address 1 1 0 W Begin write 1 1 1 W Begin read 0 X X R Ready 1 X X R Busy Cautionary note: The non-volatile memory interface is also available for access by the Add-On interface. While simultaneous accesses to the nv memory by both the Add-On and PCI are supported, via arbitration logic, software must be designed to prevent the possibility of data corruption within the memory and to provide for accurate data retrieval.
28	nvRAM Access Failed. It will indicate that the last nvRAM access has failed. This flag is cleared automatically upon the start of the next read/write operation.
27	Mailbox Flag Reset. Writing a one to this bit causes all mailbox status flags to become reset (EMPTY). It is not necessary to write this bit to 0 afterwards because it is used internally to produce a reset pulse. Since reading this bit will always return a 0, this bit is write only.
26	Reserved. Always zero.
25	Read FIFO Reset. Writing a one to this bit causes the read FIFO to reset (empty). It is not necessary to write a 0 to this bit. This bit is write only. This feature is intended for test only. It can only be asserted when the PCI is not performing any Pass-Thru accesses.
24	Reserved. Always zero.
23:16	Non-volatile Memory Address/Data Port. This 8-bit field is used in conjunction with bits 31, 30 and 29 of this register to access the external non-volatile memory. The contents written are either low address, high address, or data as defined by bits 30 and 29. This register will contain the external non-volatile memory data when the proper read sequence for bits 31 through 29 is performed.
15:12	BIST Condition Code. This field is directly connected to the PCI configuration self-test register. Bit 15 through 12 maps with the BIST register bits 3 through 0, respectively.
11:0	Reserved. Always zero.

ADD-ON PASS-THRU CONFIGURATION REGISTER (APTCR)

Register Name: Pass-Thru Configuration Register

Add-On Address: 60h

Power-up value: 80808080h

Add-On Attribute: Read/Write

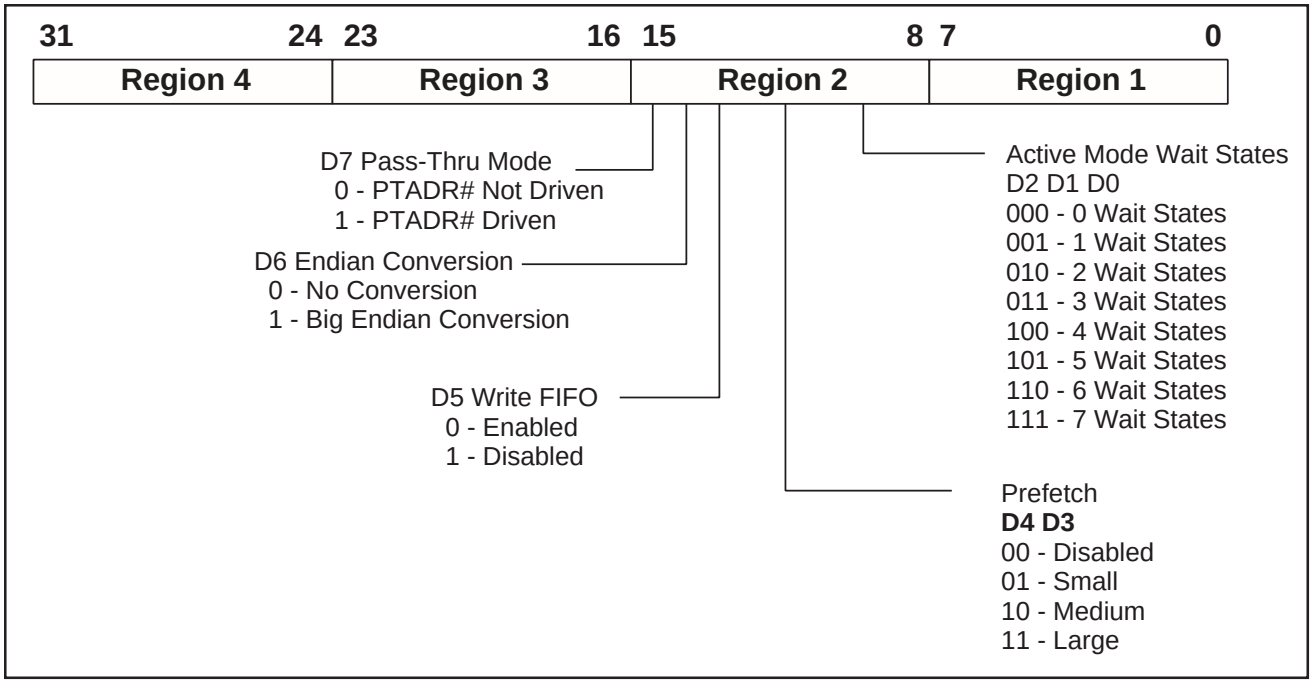
Size: 32 bits

This register controls the configuration for Pass-Thru Regions 1-4.

Byte 0 Controls Pass-Thru Region 1
Byte 1 Controls Pass-Thru Region 2
Byte 2 Controls Pass-Thru Region 3
Byte 3 Controls Pass-Thru Region 4

IMPORTANT NOTE: This register (APTCR) is physically the same as the PCI Pass-Thru configuration register (PTCR). It is intended that either the PCI system or local Add-On interface will write to this register, but not both. However, in the event that both the PCI and Add-On write to this register, whichever side wrote last wins. This register is also intended to be initialized once, prior to any PCI bus operations. Pass-Thru operation cannot be guaranteed if this register is updated while a Pass-Thru transaction is being performed.

Figure 10. Pass-Thru Configuration Register



The following describes one of the four configuration registers. All four region configuration registers are exactly the same.

Table 10. Pass-thru Configuration Register

Bit	Description
7	PTADR# mode. This bit is only valid in Active mode. If this bit is 0, PTADR# is not driven at the beginning of a Active cycle. If this bit is set to 1 (default state), the S5920 will assert PTADR# for one clock cycle after PTATN# is asserted. The Pass-Thru address is also driven while PTADR# is low. This bit is a don't care if the device is operating in Passive mode
6	Endian conversion. If this bit is set to one, the S5920 will convert the Add-On bus from the default little endian format to a big endian format.
5	Write FIFO disabled. If this bit is set to 1, the S5920 will not accept the next piece of data (on a PCI write) until the Add-On has accepted the previous piece of data. If this bit is set to 0, the S5920 will accept data from the PCI until the Pass-Thru write FIFO is full.
4:3	Prefetch. These bits control the number of DWORDs that the S5920 will prefetch after the current PCI Pass-Thru read completes. The actual amount of data prefetched depends upon any number of different scenarios. The prefetch values of "small," "medium" and "large" are available to tune the system to achieve best overall performance (i.e. optimize PCI bus transfers or optimize Add-On bus transfers). The Pass-Thru read FIFO can be enabled to prefetch in either Active mode or Passive mode.
2:0	Wait states. In Active mode, the user can program the number of wait states required by the Add-On bus to complete a transaction. Up to 7 wait states can be programmed (per region). The S5920 will count the number of clocks programmed into this register before finishing the current data transaction if PTRDY# is high. If PTRDY# is driven low, additional wait states may be inserted. Bits 2, 1 and 0 are don't care if operating in Passive mode.

INTRODUCTION

All PCI bus agents and bridges are required to implement PCI Configuration Registers. When multiple PCI devices are present, these registers must be unique to each device in the system. The specified PCI procedure for uniquely selecting a device's configuration space involves a dedicated signal, called IDSEL, connected to each motherboard PCI bus device and PCI slot.

After reset, the host executes configuration cycles to each device on the PCI bus. The configuration registers provide information on PCI agent operation and memory or I/O space requirements. These allow the PCI BIOS to enable the device and locate it within system memory or I/O space.

After a PCI reset, the S5920 can be configured for a specific application by downloading device setup information from an external non-volatile memory into the device Configuration Registers. In order to use the Pass-Thru regions, the S5920 must be used with an external nvRAM boot device If no nvRAM is used, the Base-Address Regions are disabled. However, the mailboxes and other PCI/Add-on Operation Registers can still be used (as Base-Address Region #0 comes up in its default state, defining a 128-byte I/O region).

To configure the S5920, 64 bytes of setup information are required. The rest of the boot device can be used to implement an expansion BIOS, if desired. Some of the setup information is used to initialize the S5920 PCI Configuration Registers, while other information is used to define S5920 special operating modes.

PCI RESET

Immediately following the assertion of the PCI RST# signal, the Add-On reset output SYSRST# is asserted. The Add-On reset output (SYSRST#) can be used to initialize external state machines, reset Add-On microprocessors, or other Add-On logic devices.

All S5920 Operation Registers and Configuration Registers are initialized to their default states at reset. The default values for the Configuration Registers will be overwritten by the contents of the external nv boot memory during device initialization. Configuration accesses by the host CPU while the S5920 is loading configuration will produce PCI bus retries until one of the following events occurs:

- The S5920 identifies that there is no valid boot memory (and default Configuration Register values are used).
- The S5920 finishes downloading all configuration information from a valid boot memory.

LOADING THE SERIAL NV MEMORY

Serial nv memory data transfers are performed through a two-wire, bi-directional data transfer protocol as defined by commercial serial EEPROM offerings. These devices have the advantages of low pin counts, small package size, and economical price.

A serial nv memory is initially considered valid if the first serial accesses contain the correct per-byte acknowledgments (see Figure 5). If the serial per-byte acknowledgment is not observed, the S5920 determines that no external serial nv memory is present and the AMCC default Configuration Register values specified in the PCI Configuration Register Chapter are used. Please note that the Pass-Thru interface will not operate unless a valid nv memory has been read.

The serial nvRAM is first accessed at location 0040h followed by a read to location 0041h. If either of these accesses contain anything other than FFh, the next four accesses are to locations 0050h, 0051h, 0052h and 0053h. At these locations, the data must be 80h (or 81h or 82h), FFh, E8h, and 10h, respectively, for the external nv memory to be considered valid. Once a valid external nv memory has been recognized, it is read, sequentially from location 040h to 07Fh. The data is loaded into the appropriate PCI configuration register. Some of the boot device data is not downloaded into the Configuration Registers, but is used instead to initialize some S5920 modes of operation (location 0045h, for instance). Upon completion of this sequence, the boot load terminates and PCI configuration accesses to the S5920 are acknowledged with the PCI Target Ready (TRDY#) output.

Table 1 lists the required nv memory contents for a valid configuration nv memory device.

Two pins are used to transfer data between the S5920 PCI controller and the external serial memory: a serial clock pin, SCL, and a serial data pin, SDA. The serial clock pin is an open drain output from the S5920, and the serial data pin is open drain bi-directional. The serial clock is derived by dividing the PCI bus clock by 293. This means the frequency of the serial clock is approximately 114 KHz for a 33 MHz PCI bus clock.

Note in Figure 1, a 4.7k pull-up is required on the SDA and SCL lines. During boot-up, the S5920 will only communicate with an EEPROM that has its address pins set to 0 (A[2:0] = “000). When not accessing the external nvRAM, the S5920 will tri-state the SCL and SDA signals so other two-wire serial devices can use the bus. The system designer must guarantee that the-two wire serial bus is idle whenever the S5920 wants to start an access. The S5920 does NOT perform two-wire serial arbitration. It assumes that it is the only master on the bus.

Communications with the serial memory involve several clock transitions. A start event signals the beginning of a transaction and is immediately followed by an address transfer. Each address/data transfer consists of 8 bits of information followed by a 1-bit acknowledgment. When the exchange is complete, a stop event is issued. Figure 2 shows the unique relationship defining both a start and stop event. Figure 3 shows the required timing for address/data with respect to the serial clock.

For random accesses, the sequence involves one clock to define the start of the sequence, eight clocks to send the slave address and read/write command, followed by a one-clock acknowledge, and so on. Figure 4 shows the sequence for a random write access requiring 29 serial clock transitions. At the clock speed of the S5920, this corresponds to one byte of data transferred approximately every 0.25 milliseconds. Read accesses can be either random or sequential. During boot-up, all accesses from address 40h to 7Fh are sequential. As a result, it is important the nvRAM used supports the nvRAM sequential read accesses as indicated in Figure 6. Figure 5 shows the sequence for a random byte read.

To initialize the S5920 controller’s PCI Configuration Registers, the smallest serial device necessary is a 128 x 8 organization. Although the S5920 controller only requires 64 bytes, these configuration bytes must begin at the 64-byte address offset (40h through 7Fh). This offset constraint permits the configuration image to be shared with a memory containing expansion BIOS code and the necessary preamble to identify an expansion BIOS. The largest serial device which can be used is 2 Kbytes.

Table 1. Valid External Boot Memory Contents

Address	Data	Notes
0040h-0041h	not FFFFh	This is the location that the S5920 will load a customized vendor ID. (FFFFh is an illegal vendor ID.)
0050h	82h - registers to I/O 81h - memory space 80h - memory below 1 Mbytes	This is the least significant byte of the region which initializes the S5920 configuration register BADR0. A value of 81h assigns the 32 DWORD locations of the PCI operations registers into I/O space, a value of 80h defines memory space, and a value of 82h defines memory space below 1 Mbytes.
0051h	FFh	Required
0052h	E8h	Required
0053h	10h	Required

Figure 1. S5920 to nvRAM Interface

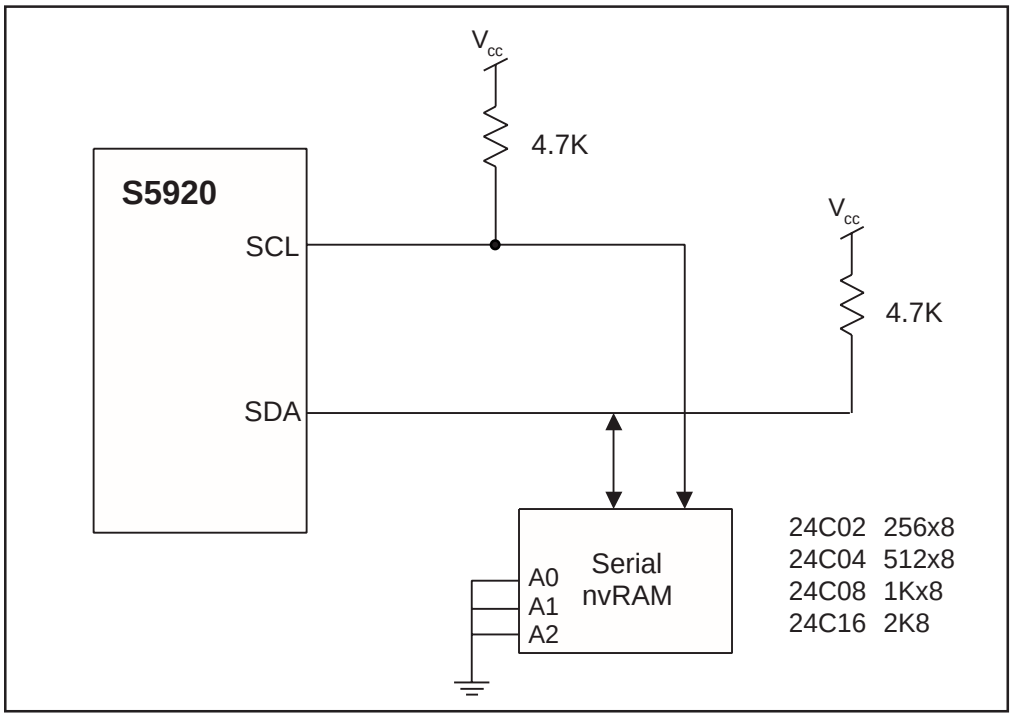


Figure 2. Serial Interface Definition of Start and Stop

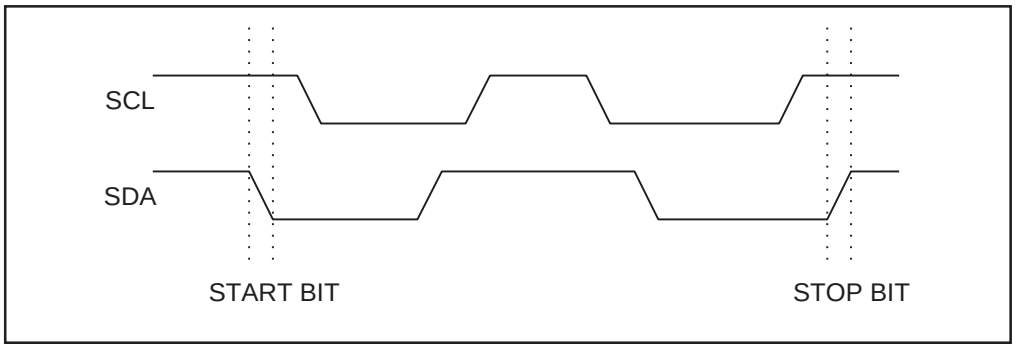


Figure 3. Serial Interface Clock Data Relationship

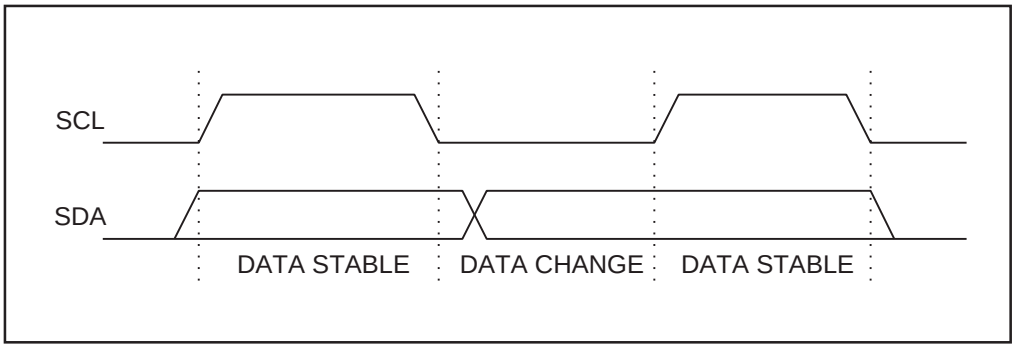


Figure 4. Serial Interface Byte Access-Write

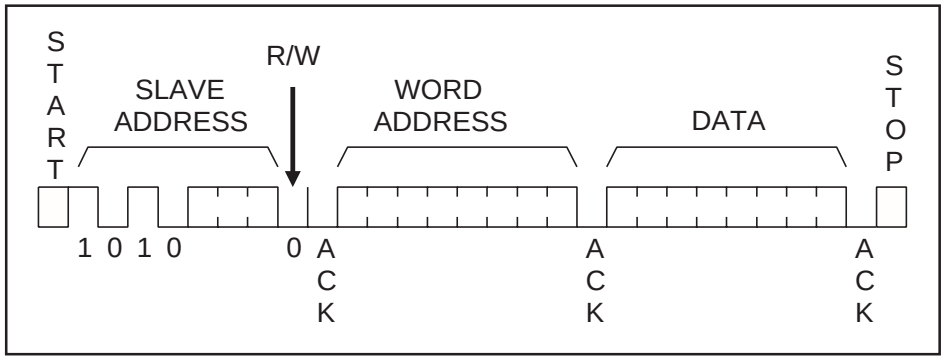


Figure 5. Serial Interface Byte Access-Read

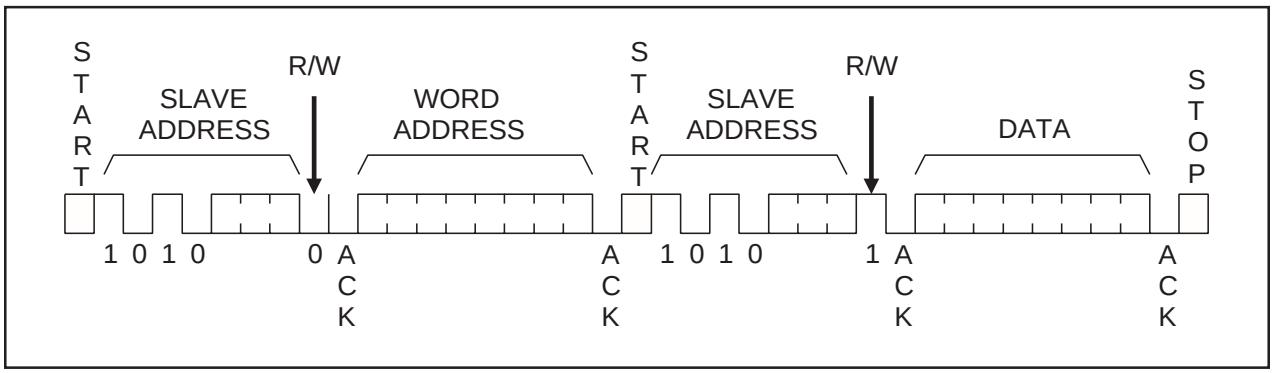


Figure 6. Serial Byte Access- Sequential Read

NON-VOLATILE MEMORY INTERFACE

The nv memory, can be accessed through the PCI interface or the Add-On interface. Accesses to the nv memory from the PCI interface are through the Reset Control Register (RCR). Accesses to the nv memory from the Add-On interface are through the Add-On Reset Control Register (ARCR).

Some nv memories can contain Expansion ROM BIOS code for use by the host CPU. During initialization, the Expansion BIOS is located within system memory. The starting location of the nv memory is stored in the Expansion ROM Base Address Register in the S5920 PCI Configuration Registers. A PCI read from this region results in the S5920 performing four consecutive byte-wide access to the nv memory device, thus assembling a complete DWORD. Writes to the nv memory are not allowed through the expansion ROM base address region. Any attempt to do so will result in data being accepted by the S5920, but simply discarded.

In the RCR and ARCR registers, bits D31:29 are command/status bits and bits D23:16 are address/data bits. These operation registers occupy the same offset (3Ch-3Fh) on their respective interfaces (Add-On or PCI). The sequence used to access the nv memory is the same in either case.

nvRAM READ/WRITE DESCRIPTION

There are four different mechanisms to access the external nvRAM:

1) During boot-up (RST# deasserted), the S5920 will automatically read out the nvRAM addresses 40h - 7Fh.
2) Via the PCI Configuration Expansion ROM Base Address Register (EXROM). This is READ-ONLY.
3) Via the PCI Reset Control Register (RCR). This is READ/WRITE.
4) Via the Add-On Reset Control Register (ARCR). This is READ/WRITE.

The boot-up sequence is a built-in function, and is affected by the contents of the nvRAM.

The Expansion ROM Base Address Register is used if expansion BIOS is stored in the external nvRAM. This register can be enabled for a 2K memory size, and is mapped to access the contents of the nvRAM. When a read is performed to an address in the range of the EXROM base address, a read sequence is started to the nvRAM. As this sequence is extremely slow, the PCI will be greeted with a Retry. Meanwhile, the nvRAM interface circuitry will be performing four sequential byte accesses to the nvRAM at the offset indicated by the PCI address. For example, if the EXROM Base Address is programmed with 100000h, and the PCI performs a read to address 100040h, this will initiate a read from address 40h of the nvRAM. Once addresses 40h, 41h, 42h and 43h have been read and stored in the nvRAM interface, the S5920 is ready to provide the data to the original PCI device requesting the data. Once the original master comes back to read the data (which it should, as it received a Retry to its initial read), it will get a TRDY# along with the 4 bytes of data that were read from the nvRAM. If the master comes back to retry the read, but the nvRAM interface is not finished with its accesses, the master will again be greeted with a Retry. If a master attempts to read from a different EXROM address, it will also be greeted with a Retry. Only a read with the original address (in our example, a read to address 100040h) will allow the transaction to complete. As a result, if the original master never comes back to Retry the read, the EXROM interface will be hung. Only other EXROM accesses will be hung, as the nvRAM interface will still be operational via the PCI's RCR and the Add-On's ARCR.

Accesses to the nvRAM via the PCI's Reset Control Register (RCR) are a bit more involved for the programmer. There are 12 bits of this register that perform both reads and writes. Bits 23-16 to provide Address/Data information, bits 31-29 are used to provide control information, and bit 28 indicates whether the nvRAM access was successful or not. The control bits 31-29 are assigned as follows (where W/R indicates the type of PCI access to the RCR):

D31	D30	D29	W/R	nvRAM Interface Function
0	X	X	W	Inactive
1	0	0	W	Load low address byte
1	0	1	W	Load high address byte
1	1	0	W	Begin write
1	1	1	W	Begin read
0	X	X	R	Ready
1	X	X	R	Busy

These control bits are used along with the Address/Data bits 23-16 to configure the type of nvRAM operation (read or write), the address being accessed, and a place to store the write data or the data read from the nvRAM. One can interface with this register in either byte-wide or word-wide fashion. For a word-wide access, the command (bits 31-29) and Address/Data (bits 23-16) are written to the RCR with one PCI write. For a byte-wide access, the command (bits 31-29) is written first, followed by the Address/Data (bits 23-16). This takes two PCI transfers.

When performing a byte-wide RCR access, users need to write the command indicating how the data is to be used, followed by the data. These commands will assert the internal signals LOAD_LOW_ADDR, LOAD_HIGH_ADDR or LOAD_WR_DATA. Only one signal is asserted at any time: once one is asserted, the others are deasserted.

The final read/write interface to the external nvRAM is via the Add-On Reset and Control Register (ARCR). This mechanism is identical to that used for the PCI's RCR, except that the Add-On interface is used to access the nvRAM via the ARCR. The latency is a bit longer as well, due to the synchronization that must be performed between the Add-On clock and the PCI clock.

While on-chip arbitration logic allows simultaneous accesses to the nvRAM via the PCI's RCR and Add-On's ARCR (by queuing up the commands), there is no logic to prevent each interface from overwriting nvRAM contents. If an interface writes to a memory location that the other interface has already has written to, the value at that location will be overwritten.

What follows are the sequence of steps required to access the nvRAM via the RCR. All the scenarios assume that the RCR is being controlled via PCI bus transactions. By replacing RCR with ARCR in the examples below, the operations are identical for an Add-On device.

The following sequence is used to perform nvRAM writes when accessing the RCR/ARCR in a byte-wide fashion:

1) Verify that busy bit, RCR(31), is not set by reading RCR(31). If set, hold off starting the write sequence (repeat step 1 until this bit clears).

2) Write to RCR(31:29) = "100", the command to load the low address byte. This will assert the internal signal LOAD_LOW_ADDR, which is used to enable the loading of the low-address register (NVRAM_LOW_ADDR).

3) Write to RCR(23:16) with the low address byte. Since signal LOAD_LOW_ADDR is asserted, the data will be written to register NVRAM_LOW_ADDR. As long as LOAD_LOW_ADDR is asserted, a write to RCR(23:16) will continue to overwrite register NVRAM_LOW_ADDR.

4) Write to RCR(31:29) = "101", the command to load the high address byte. This will assert the internal signal LOAD_HIGH_ADDR, which is used to enable the loading of the high-address register (NVRAM_HIGH_ADDR).

5) Write to RCR(23:16) with the high address byte. Since signal LOAD_HIGH_ADDR is asserted, the data will be written to register NVRAM_HIGH_ADDR. Note that as the nvRAM address is limited to 11 bits, only the 3 lsbs of this write data is actually used. As long as LOAD_HIGH_ADDR is asserted, a write to RCR(23:16) will continue to overwrite register NVRAM_HIGH_ADDR.

6) Write to RCR(31:29) = "000", a dummy command to deassert either LOAD_LOW_ADDR or LOAD_HIGH_ADDR (whichever occurred last), and to assert internal signal LOAD_WR_DATA. This signal is used to enable the loading of the write data register. LOAD_WR_DATA will remain asserted until another command is issued (load low/high address, begin read/write). As long as LOAD_WR_DATA is asserted, a write to RCR(23:16) will continue to overwrite the write data register.

7) Write to RCR(23:16) the byte to be written. Since the signal LOAD_WR_DATA is asserted, the data will be written to the write data register.

8) Write to RCR(31:29) = "110", the command to start the nvRAM write operation. This will lead to the deassertion of LOAD_WR_DATA and will set the busy bit, RCR(31). The nvRAM interface controller will now initiate a write operation with the external nvRAM.

9) Poll the busy bit until it is no longer set. Once cleared, it is now safe to perform another write/read operation to the external nvRAM. The XFER_FAIL flag (bit 28) can be used to determine whether the transfer was successful or not. If XFER_FAIL is asserted, this indicates that a transfer to the nvRAM did not receive an ACKNOWLEDGE, and the write transfer should not be considered successful. This flag remains set until the start of the next read/write operation.

The busy bit will remain set until the nvRAM interface has completed writing the data byte to the external nvRAM, and has verified that the write sequence is finished. The nvRAM "shuts down" during a write and will not accept any new commands (does not generate an ACKNOWLEDGE) until it finishes the write operation. The S5920 will continue to send commands to the nvRAM until it responds with an ACKNOWLEDGE, after which it clears the busy bit, indicating that the write operation is truly complete. If the busy bit were to be cleared after the nvRAM interface finished the write, but before the external nvRAM was actually finished, a scenario exists where a successive write would be ignored. In this case, the software driver could not use the busy bit to determine when to start a new write, but

would need to insert a delay (determined by the "shut down" time of the nvRAM, between 5-10 ms). Fortunately, the S5920 implements the Acknowledge Polling scheme described above, which will not take away the busy bit until the write is truly finished, and the external nvRAM is available for accesses.

The following sequence is used to perform nvRAM writes when accessing the RCR/ARCR in a byte-wide fashion:

1) Verify that busy bit, RCR(31), is not set by reading RCR(31). If set, hold off starting the read sequence (repeat step 1 until this bit clears).

2) Write to RCR(31:29) = "100", the command to load the low address byte. This will assert the internal signal LOAD_LOW_ADDR, which is used to enable the loading of the low-address register (NVRAM_LOW_ADDR).

3) Write to RCR(23:16) with the low address byte. Since signal LOAD_LOW_ADDR is asserted, the data will be written to the register NVRAM_LOW_ADDR. As long as LOAD_LOW_ADDR is asserted, a write to RCR(23:16) will continue to overwrite register NVRAM_LOW_ADDR.

4) Write to RCR(31:29) = "101", the command to load the high address byte. This will assert the internal signal LOAD_HIGH_ADDR, which is used to enable the loading of the high-address register (NVRAM_HIGH_ADDR).

5) Write to RCR(23:16) with the high address byte. Since signal LOAD_HIGH_ADDR is asserted, the data will be written to the register NVRAM_HIGH_ADDR. Note that as the nvRAM address is limited to 11-bits, only the 3-lsb's of this write data is actually used. As long as LOAD_HIGH_ADDR is asserted, a write to RCR(23:16) will continue to overwrite register NVRAM_HIGH_ADDR.

6) Write to RCR(31:29) = "111", the command to start the nvRAM read operation. This will set the busy bit, RCR(31). The nvRAM interface controller will now initiate a read operation to the external nvRAM.

7) Poll the busy bit until it is no longer set. Once cleared, the read data will be located in RCR(23:16). In addition, evaluate the XFER_FAIL flag (bit 28) to determine whether the transfer was successful or not. If XFER_FAIL is asserted, this indicates that a transfer to the nvRAM did not receive an ACKNOWLEDGE, and the read data in RCR(32:16) may not be valid. This flag remains set until the start of the next read/write operation.

When performing a word/double-word RCR access, you can combine the data and control in the same command. The following is the sequence for a write:

1) Verify that busy bit, RCR(31), is not set by reading RCR(31). If set, hold off starting the write sequence (repeat step 1 until the bit clears).

2) Write to RCR(31:29) = "100" and RCR(23:16) with the low address byte. This will directly load NVRAM_LOW_ADDR with RCR(23:16).

3) Write to RCR(31:29) = "101" and RCR(23:16) with the high address byte. This will directly load NVRAM_HIGH_ADDR with RCR(23:16).

4) Write to RCR(31:29) = "110" and RCR(23:16) with the write data. This will directly load the write data register with RCR(23:16). This will also set the busy bit, RCR(31). The nvRAM interface controller will now initiate a write operation to the external nvRAM.

5) Poll the busy bit until it is no longer set. Once cleared, it is now safe to perform another write/read operation to the external nvRAM. In addition, evaluate the XFER_FAIL flag (bit 28) to determine whether the transfer was successful or not. If XFER_FAIL is asserted, this indicates that a transfer to the nvRAM did not receive an ACKNOWLEDGE. The write should not be considered successful. This flag remains set until the start of the next read/write operation.

The following sequence is used for a read:

1) Verify that the busy bit, RCR(31), is not set by reading RCR(31). If set, hold off starting the read sequence (repeat step 1 until the bit clears).

2) Write to RCR(31:29) = "100" and RCR(23:16) with the low address byte. This will directly load NVRAM_LOW_ADDR with RCR(23:16).

3) Write to RCR(31:29) = "101" and RCR(23:16) with the high address byte. This will directly load NVRAM_HIGH_ADDR with RCR(23:16).

4) Write to RCR(31:29) = "111". This will set the busy bit, RCR(31). The nvRAM interface controller will now initiate a read operation with the external nvRAM.

5) Poll the busy bit until it is no longer set. Once cleared, the read data will be located in RCR(23:16). In addition, evaluate the XFER_FAIL flag (bit 28) to determine whether the transfer was successful or not. If XFER_FAIL is asserted, this indicates that a

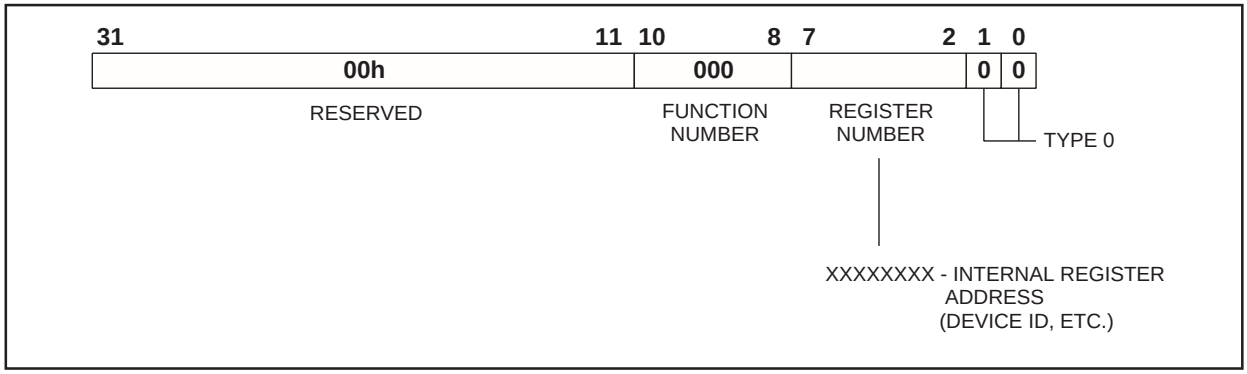
transfer to the nvRAM did not receive an ACKNOWLEDGE. The read data in RCR(32:16) should not be considered valid. This flag remains set until the start of the next read/write operation.

PCI BUS CONFIGURATION CYCLES

Cycles beginning with the assertion IDSEL and FRAME# along with the two configuration command states for C/BE[3:0] (configuration read or write) access the selected device's configuration space. During the address phase of the configuration cycle just described, the values of AD0 and AD1 identify if the access is a Type 0 configuration cycle or a Type 1 configuration cycle. Type 0 cycles have AD0 and AD1 equal to 0 and are used to access PCI bus agents. Type 1 configuration cycles are intended only for bridge devices and have AD0 as a 1 with AD1 as a 0 during the address phase.

The S5920 PCI device is a bus agent (not a bridge) and responds only to a Type 0 configuration accesses. Figure 7 depicts the state of the AD bus during the address phase of a Type 0 configuration access. The S5920 controller does not support the multiple function numbers field (AD[10:8]) and only responds to the all-0 function number value.

Figure 7. PCI AD Bus Definition Type 0 Configuration Access



The configuration registers for the S5920 PCI controller can only be accessed under the following conditions:

- IDSEL high (PCI slot unique signal which identifies access to configuration registers) along with FRAME# low.
- Address bits A0 and A1 are 0 (Identifies a Type 0 configuration access).
- Address bits A8, A9, and A10 are 0 (Function number field of 0 supported).
- Command bits, C/BE[3:0]# must identify a configuration cycle command (101X).

Figure 8 describes the signal timing relationships for configuration read cycles. Figure 9 describes configuration write cycles.

Figure 8. Type 0 Configuration Read Cycles

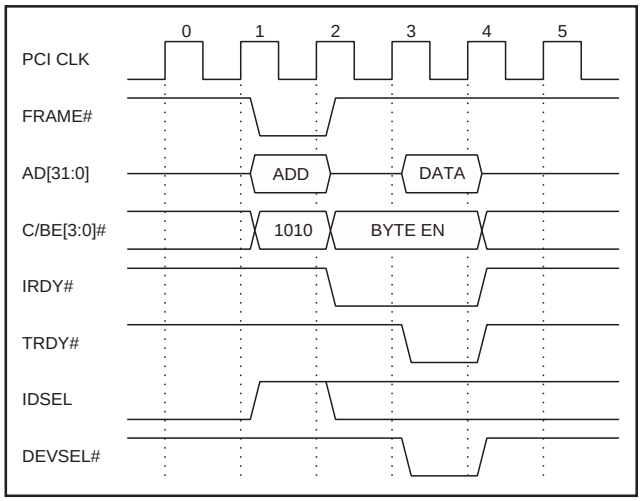
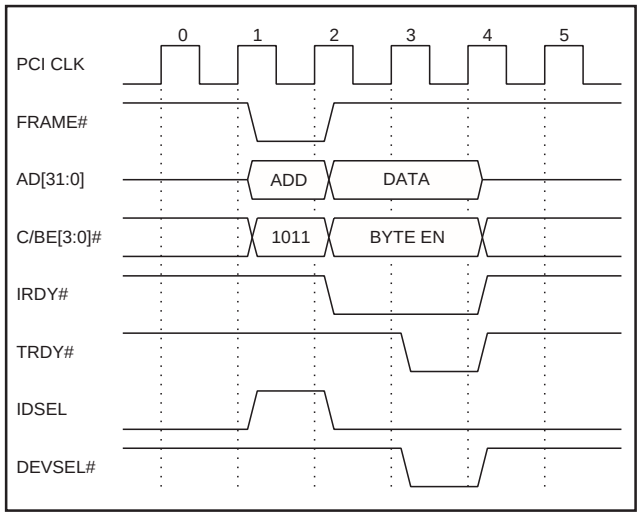


Figure 9. Type 0 Configuration Write Cycles



EXPANSION BIOS ROMS

This section provides an example of a typical PC-compatible expansion BIOS ROM. Address offsets 040h through 07Fh represent the portion of the external nv memory used to boot-load the S5920 controller.

Whether the expansion ROM is intended to be executable code is determined by the contents of the first three locations (starting at offset 0h) and a byte checksum over the defined length. The defined length is specified in the byte at address offset 0002h. Table 2 lists each field location by its address offset, its length, its value, and description.

Table 2. PC Compatible Expansion ROM

Byte Offset	Byte Length (decimal)	Binary Value	Description	Example
0h	1	55h	BIOS ROM signature byte 1	55h
1h	1	AAh	BIOS ROM signature byte 2	AAh
2h	1	variable	Length in multiples of 512 bytes	01h
3h	4	variable	Entry point for INIT function.	
7h-17h	17	variable	Reserved (application unique data)	
18h-19h	2	variable	Pointer to PCI Data Structure	
1Ah-3Fh	38	variable	user-defined	

The following represents the boot-load image for the S5920 controller's PCI configuration register:

Table 2. PC Compatible Expansion ROM (Continued)

Byte Offset	Byte Length (decimal)	Binary Value	Description	Example
40h	2	Vendor ID	(see page 2-23)	10E8h
42h	2	Device ID	(see page 2-24)	5920h
44h	1	not used		xxh
45h	1	S5920 Special Modes	(see page 2-89 and 2-129)	01h
46h	2	not used		xxxxh
48h	1	Revision ID	(see page 2-29)	00h
49h	3	class code	(see page 2-30)	FF0000h
4Ch	1	not used		xxh
4Dh	1	not used		xxh
4Eh	1	your header type	(see page 2-37)	00h
4Fh	1	self-test, if desired	(see page 2-38)	80h or 00h
50h	1	80h, 81h or 82h	(required, see page 2-39, and page 2-74, Table 1)	80h, 81h or 82h
51h	1	FFh	(required per Table 1)	FFh
52h	1	E8h	(required per Table 1)	E8h
53h	1	10h	(required per Table 1)	10h
54h	4	base addr. #1	(see page 2-39)	00000000h
58h	4	base addr. #2	(see page 2-39)	00000000h
5Ch	4	base addr. #3	(see page 2-39)	00000000h
60h	4	base addr. #4	(see page 2-39)	00000000h
64h	4	not used		00000000h
68h	8	not used		XXh
6Ch	2	SVID	(see page 2-44)	5555h
6Eh	2	SID	(see page 2-45)	3333h
70h	4		[Expansion ROM base addr.] (see page 2-46) (example shows 2K bytes)	FFFFF801h
74h	8	not used		XXh
7Ch	1	Interrupt line	(see page 2-48)	0Ch
7Dh	1	Interrupt pin	(see page 2-49)	01h
7Eh	1	not used	(see page 2-48)	xxh
7Fh	1	not used	(see page 2-49)	xxh
80h — 1FFh, 2FFh			application specific	
3FFh etc.			Byte checksum, location dependent on value for length field at offset 0002h.	

A 16-bit pointer at location 18h of the PC expansion ROM identifies the start offset of the PCI data structure. The PCI data structure is shown in Table 3 and contains various vendor, product, and program descriptions. This structure is provided here for reference only - the user should refer to the PCI BIOS specification for complete details.

Note: The access time for large serial devices should be considered, since it may cause a lengthy system delay during initialization. For example, a 2 Kbytes serial device will take about 1 second to be read. Many systems, even when BIOS ROMs are ultimately shadowed into system RAM, may read this memory space twice (once to validate its size and checksum, and once to move it into RAM).

Table 3. PCI Data Structure

Byte Offset	Byte Length (decimal)	Binary Value	Description
0h	4	'PCIR'	Signature, the ASCII string 'PCIR' where 'P' is at offset 0, 'C' at offset 1, and so on
4h	2	variable	Vendor Identification
6h	2	variable	Device Identification
8h	2	variable	Pointer to Vital Product Data
Ah	2	variable	PCI Data Structure Length (starts with signature field)
Ch	1	variable	PCI Data Structure Revision (=0 for this definition)
Dh	3	variable	Class Code
10h	2	variable	Image Length
12h	2	variable	Revision Level
14h	2	variable	Code Type
15h	1	variable	Indicator (bit D7=1 signifies "last image")
16h	2	0000h	Reserved

PCI BUS INTERFACE

This section details various events which may occur on the S5920 PCI bus interface. Since the S5920 functions as a target or slave device, signal timing details are given for target transactions only.

PCI BUS TRANSACTIONS

Because the PCI bus utilizes multiplexed address/data pins (AD[31:0]), every PCI bus transaction consists of an address phase followed by a data phase. An address phase is defined as the clock period in which FRAME# transitions from inactive to active. During the address phase, a bus command is driven by the initiator on the C/BE[3:0]# signal pins. If the command indicates a PCI read, the clock cycle following the address phase is used to perform a "bus turn-around" cycle. A turn-around cycle is a clock period in which the address/data bus is not driven by an initiator or a target device. This is used to avoid PCI bus contention. For a write command, a turn-around cycle is not needed, and the bus goes directly from an address phase to a data phase.

All PCI bus transactions consist of an address phase followed by one or more data phases. During the one-PCI-clock-long address phase, the bus address and command information is latched into the S5920. The number of data phases depends on how many data transfers are desired or are possible within a given initiator-target pair. A data phase consists of at least one PCI clock. FRAME# is deasserted to indicate that the final data phase of a PCI cycle is occurring. Wait states may be added to any data phase (each wait state is one PCI clock).

The PCI bus command presented on the C/BE[3:0]# pins during the address phase can represent 16 possible states. Table 1 lists the PCI commands and those which are supported by the S5920. A "Yes" in the "Supported by S5920" column in Table 1 indicates that the S5920 device will assert the signal DEVSEL# when that particular command is issued along with the appropriate PCI address.

The completion or termination of a PCI cycle can be signaled in several ways. In most cases, the completion of the final data phase is indicated by the assertion of the ready signals from both the target (TRDY#) and initiator (IRDY#) while FRAME# is inactive. In some cases, the target is not able to continue or support a burst transfer and will assert a STOP# signal. This is referred to as a target disconnect. There is also the case where an addressed device does not exist, and the signal DEVSEL# is not driven. In this case, the initiator is responsible for ending the cycle. This is referred to as a master abort. The bus is returned to the idle phase when both FRAME# and IRDY# are deasserted.

Table 1. PCI Bus Commands

C/BE[3:0]#	Command Type	Supported
0000	Interrupt Acknowledge	No
0001	Special Cycle	No
0010	I/O Read	Yes
0011	I/O Write	Yes
0100	Reserved	No
0101	Reserved	No
0110	Memory Read	Yes
0111	Memory Write	Yes
1000	Reserved	No
1001	Reserved	No
1010	Configuration Read	Yes
1011	Configuration Write	Yes
1100	Memory Read Multiple	Yes [1]
1101	Reserved	No
1110	Memory Read Line	Yes [1]
1111	Memory Write and Invalidate	Yes [2]

1. Memory Read Multiple and Memory Read Line are executed as a Memory Read.
2. Memory Write and Invalidate is executed as a Memory Write.

PCI BURST TRANSFERS

The PCI bus, by default, expects burst transfers to be executed. To successfully perform a burst transfer, both the initiator and target must order their burst address sequence in an identical fashion. There are two different ordering schemes: linear address incrementing and 80486 cache line fill sequencing.

The S5920 supports only linear burst ordering. Attempts to perform burst transfers with a scheme other than this will cause the STOP# signal to be asserted during the first data phase, thus issuing a disconnect to the initiator. The S5920 completes the initial data phase successfully, but asserting STOP# indicates that the next access needs to be a completely new cycle.

Some accesses to the S5920 controller do not support burst transfers. For example, the S5920 does not allow burst transfers when accesses are made to the configuration or operation registers. Attempts to perform burst transfers to these regions will cause a disconnect on the PCI bus, as described above. Expansion ROM accesses also do not support bursts, and will respond in the same way. Accesses to memory or I/O regions defined by the Base Address Registers 1-4 may be bursts, if desired.

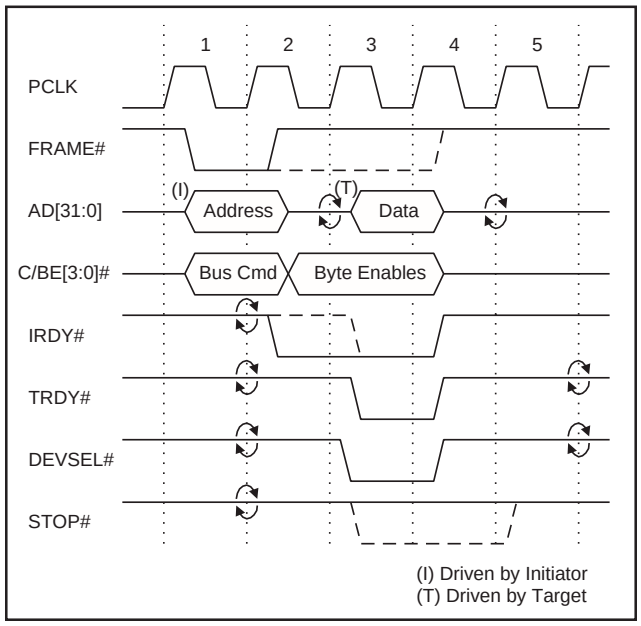
PCI READ TRANSFERS

The S5920 responds to PCI bus memory or I/O read transfers when it is selected as a target.

PCI targets may drive DEVSEL# and TRDY# after the end of the address phase. TRDY# is not driven until the target can provide valid data for the PCI read.

Read accesses from the S5920 operation registers are shown in Figure 1. The S5920 conditionally asserts STOP# in clock period 3 if the initiator keeps FRAME# asserted during clock period 2 with IRDY# asserted (indicating a burst is being attempted). Wait states may be added by the initiator by not asserting the signal IRDY# during clock 3 and beyond. If FRAME# remains asserted, but IRDY# is not asserted, the initiator is just adding wait states, not necessarily attempting a burst.

Figure 1. Single Data Phase PCI Bus Read of S5920 Registers or Expansion ROM



There are only two conditions where accesses to the S5920 do not return TRDY#, but assert STOP# instead. This condition is called a target-initiated termination or target disconnect. This can occur when a read attempt is made to an empty Pass-Thru FIFO. The second condition may occur when read accesses to the expansion ROM generate a retry if the nvRAM interface has not finished reading 4 bytes.

When burst read transfers are attempted to the S5920 operation registers, configuration registers or expansion ROM, STOP# is asserted during the first data transfer to indicate to the initiator that no further transfers (data phases) are possible. This is a target-initiated termination where the target disconnects after the first data phase. Figure 2 shows the signal relationships during a burst read attempt to the S5920 operation registers.

PCI WRITE TRANSFERS

Write transfers on the PCI bus are one clock period shorter than read transfers. This is because the AD[31:0] bus does not require a turn-around cycle between the address and data phases.

Write accesses to the S5920 operation registers are shown in Figure 3. Here, the S5920 asserts the signal STOP# in clock period 3. STOP# is asserted because the S5920 does not support burst writes to operation registers. Wait states may be added by the initiator by not asserting the signal IRDY# during clock 2 and beyond. There is only one condition where writes to S5920 internal registers do not return TRDY# (but do assert STOP#). This is called a target-initiated termination or target disconnect. This occurs when a write attempt is made to a full Pass-Thru FIFO. The assertion of STOP# without the assertion of TRDY# indicates that the initiator should retry the operation later. The S5920 will sustain a burst as long as the FIFO is not full.

Figure 2. Burst PCI Bus Read Attempt to S5920 Registers or Expansion ROM

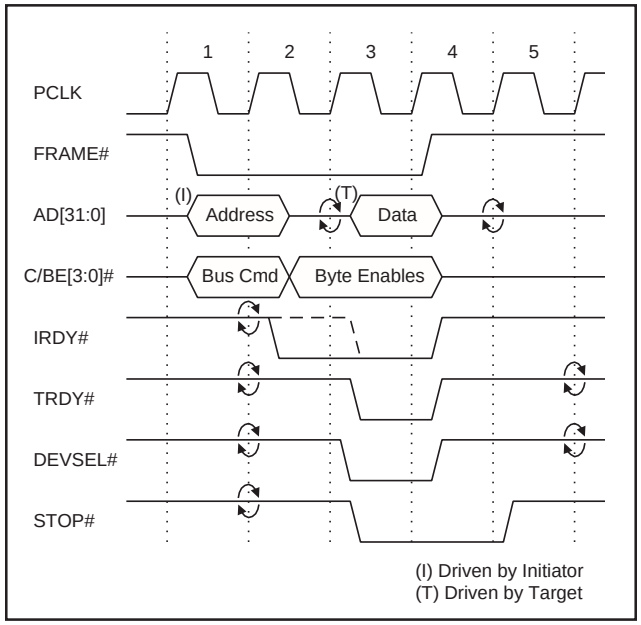
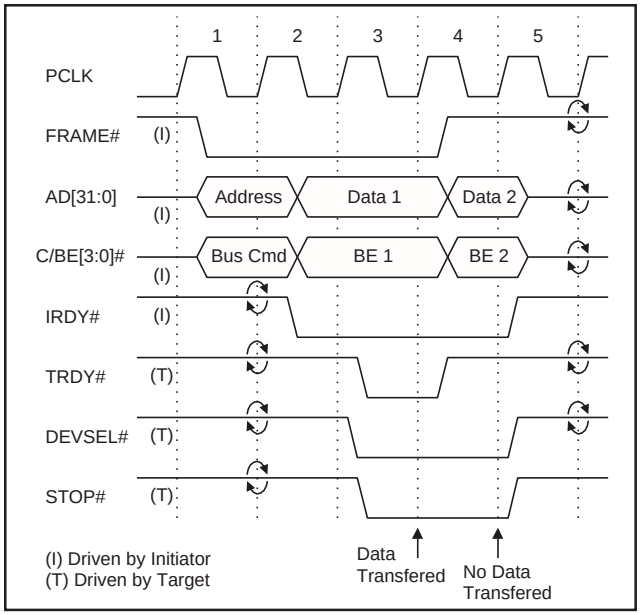


Figure 3. Burst PCI Bus Write of S5920 Registers



Target-Initiated Termination

There are situations where the target may end a transfer prematurely. This is called "target-initiated termination." Target termination falls into three categories: disconnect, retry, and target abort. Only the disconnect termination completes a data transfer.

Target Disconnects

There are many situations where a target may disconnect. Slow responding targets may disconnect to permit more efficient (faster) devices to be accessed while they prepare for the next data phase. Or a target may disconnect if it recognizes that the next data phase in a burst transfer is out of its address range. A target disconnects by asserting STOP#, TRDY#, and DEVSEL# as shown in Figures 4a and 4b. The initiator in Figure 4a responds to the disconnect condition by deasserting FRAME# on the following clock but does not complete the data transfer until IRDY# is asserted. The timing diagram in Figure 4b also applies to the S5920.

The S5920 performs a target disconnect if a burst access is attempted to any of its PCI Operation/ Configuration Registers, or to the Expansion ROM.

Target Requested Retries

The S5920 initiates a retry for Pass-Thru writes when the Write FIFO is full, and for Pass-Thru reads when the Add-On cannot supply data within 16 PCI clocks from the assertion of FRAME# (for the first data phase of a burst). A retry is requested by a target by asserting both STOP# and DEVSEL# while TRDY# is deasserted. Figure 5 shows the behavior of the S5920 when performing a target-initiated retry.

Figure 4a. Target Disconnect Example 1 (IRDY# Deasserted)

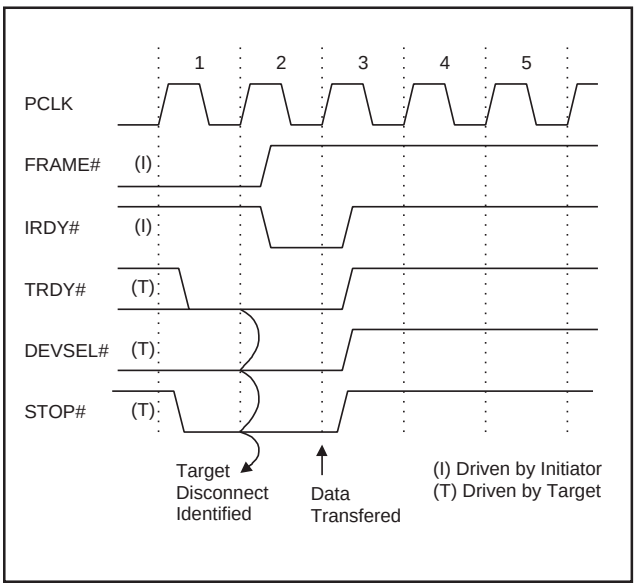
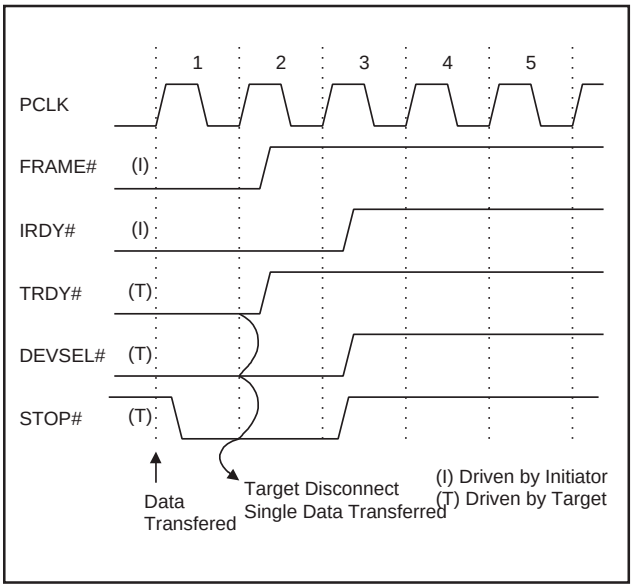


Figure 4b. Target Disconnect Example 2 (IRDY# Asserted)



Target Aborts

A target abort termination represents an error condition when no number of retries will produce a successful target access. A target abort is uniquely identified by the target deasserting DEVSEL# and TRDY# while STOP# is asserted. When a target performs an abort, it must also set bit 11 of its PCI Status register (PCISTS). The S5920 never responds with a target abort when accessed. Target termination types are summarized in Table 2.

Target Latency

The PCI specification requires that a selected target relinquish the bus should an access to that target require more than eight PCI clock periods (16 clocks for the first data phase, 8 clocks for each subsequent data phase). This prevents slow target devices from potentially monopolizing the PCI bus and also allows more accurate estimations for bus access latency.

Note that a special mode is available to the user which will allow for this mechanism to be disabled, thus violating the PCI 2.1 Specification. If a value of 0 is programmed into the serial nvRAM location 45h, bit 0, target latency is ignored. In this case, the S5920 will never issue a retry/disconnect in the event of a slow Add-On device. This programmable bit is only provided for flexibility, and most users should leave this bit set to 1.

nvRAM Location 45h, bit 0 = 0 : No disconnect for slow Add-On device.

nvRAM Location 45h, bit 0 = 1 : PCI 2.1 compliant

Target Locking

It is possible for a PCI bus master to obtain exclusive access to a target ("locking") through use of the PCI bus signal LOCK#. LOCK# is different from the other PCI bus signals because its ownership may belong to any bus master, even if it does not currently have ownership of the PCI bus. The ownership of LOCK#, if not already claimed by another master, may be achieved by the current PCI bus master on the clock period following the initial assertion of FRAME#. Figure 6 describes the signal relationship for establishing a lock. The ownership of LOCK#, once established, persists even while other bus masters control the bus. Ownership can only be relinquished by the master which originally established the lock.

PCI Bus Access Latency Components

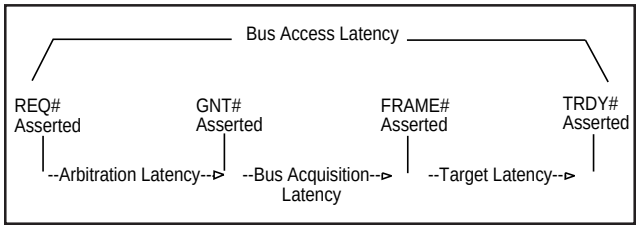


Figure 5. Target-Initiated Retry

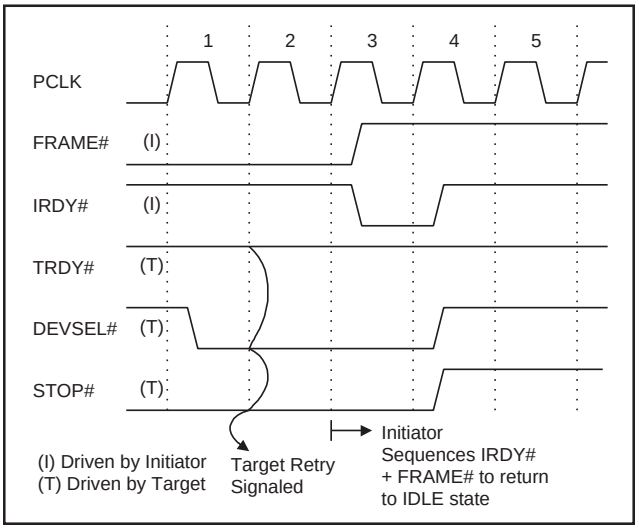


Figure 6. Engaging the LOCK# Signal

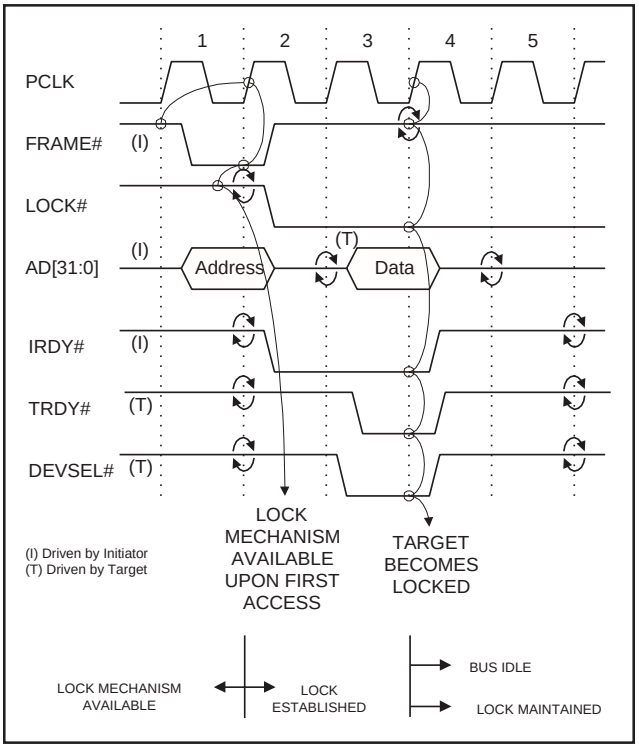


Table 2. Target Termination Type

Termination	DEVSEL#	STOP#	TRDY#	Comment
Disconnect	on	on	on	Data is transferred. Transaction needs to be re-initiated to complete.
Retry	on	on	off	Data was not transferred. Transaction should be tried later.
Abort	off	on	off	Data was not transferred. Fatal error.

Targets selected with LOCK# deasserted during the assertion of FRAME# (clock period 1 of Figure 6), which encounter the assertion of LOCK# during the following clock (clock period 2 of Figure 6) are thereafter considered “locked.” A target, once locked, requires that subsequent accesses to it deassert LOCK# while FRAME# is asserted. Figure 7 shows a subsequent access to a locked target by the master which locked it. Because LOCK# is owned by a single master, only that master is able to deassert it at the beginning of a transaction (assuming successful access to the locked target). A locked target can only be unlocked during the clock period following the last data transfer of a transaction when the LOCK# signal is deasserted.

An unlocked target ignores LOCK# when it observes that LOCK# is already asserted during the first clock period of a transaction. This allows other masters to access other (unlocked) targets. If an access to a locked target is attempted by a master other than the one that locked it, the target responds with a retry request, as shown in Figure 8.

The S5920 responds to and supports bus masters which lock it as a target.

PCI BUS INTERRUPTS

The S5920 controller is able to generate PCI bus interrupts by asserting the PCI bus interrupt signal (INTA#). INTA# is a multi-sourced, wire-ORed signal on the PCI bus and is driven by an open drain output on the S5920. The assertion and deassertion of INTA# have no fixed timing relationship with respect to the PCI bus clock. Once the S5920 asserts INTA#, it remains asserted until the interrupt source is cleared by a write to the Interrupt Control/Status Register (INTCSR). In the case of the external Add-On Interrupt, INTA# will remain set as long as the ADDINT# pin is driven low by an Add-On device(s). The source(s) driving ADDINT# must deassert this input before the PCI interrupt (INTA#) is driven to the false state. It is the responsibility host software to clear the Add-On interrupt source before exiting its interrupt handler routine.

PCI BUS PARITY ERRORS

The PCI specification defines two error-reporting signals, PERR# and SERR#. These signals indicate a parity error condition on the signals AD[31:0], C/BE[3:0]#, and PAR. The validity of the PAR signal is delayed one clock period from its corresponding AD[31:0] and C/BE[3:0]# signals. Even parity is supported by PCI: when the total number of ones in the group of signals AD[31:0] and C/BE[3:0]# is equal to an even number the parity bit will be deasserted. If an odd number of ones is seen, the parity bit will be asserted.

PERR# is the error-reporting mechanism for parity errors that occur during the data phase for all but PCI Special Cycle commands. SERR# is the error-reporting mechanism for parity errors that occur during the address phase.

The timing diagram in Figure 9 shows the timing relationships between the signals AD[31:0], C/BE[3:0]#, PAR, PERR# and SERR#.

The S5920 asserts SERR# if it detects odd parity during an address phase, if enabled. The SERR# enable bit is bit 8 in the S5920 PCI Command Register (PCICMD). The odd parity error condition involves the state of signals AD[31:0] and C/BE[3:0]# when FRAME# is first asserted and the PAR signal during the following clock. If an error is detected, the S5920 asserts SERR# on the following (after PAR valid) clock. Since many targets may observe an error on an address phase, the SERR# signal is an open-drain multi-sourced, wire-ORed signal on the PCI bus. The S5920 drives SERR# low for one clock period when an address phase error is detected. Once an SERR# error is detected by the S5920, the PCI Status register bit 14, System Error, is set and remains set until cleared through software or a hardware reset.

The PERR# signal is similar to the SERR# with two differences: it reports errors for the data phase and is only asserted by the device receiving the data. The S5920 drives this signal (removed from tri-state) when it is the selected target for write transactions. The parity error conditions are only reflected by the PERR# pin if the Parity Error Enable bit (bit 6) of the PCI Command Register is set. Upon the detection of a data parity error, the Detected Parity Error bit (bit 15) of the PCI Status Register is set (PCISTS). Unlike the PERR# signal pin, this Status bit is set regardless of the state of the PCI Command Register's Parity Error Enable bit.

The assertion of PERR# occurs two clock periods following the data transfer. This two-clock delay occurs because the PAR signal does not become valid until the clock following the transfer, and an additional clock is provided to generate and assert PERR# once an error is detected. PERR# is only asserted for one clock cycle for each error sensed. The S5920 only qualifies the parity error detection during the actual data transfer portion of a data phase (when both IRDY# and TRDY# are asserted).

Figure 7. Access to a Locked Target by its Owner

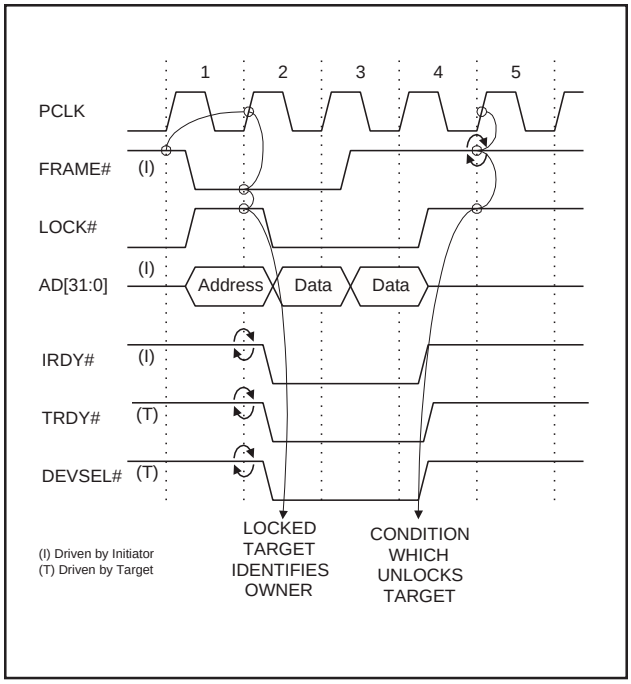


Figure 8. Access Attempt to a Locked Target

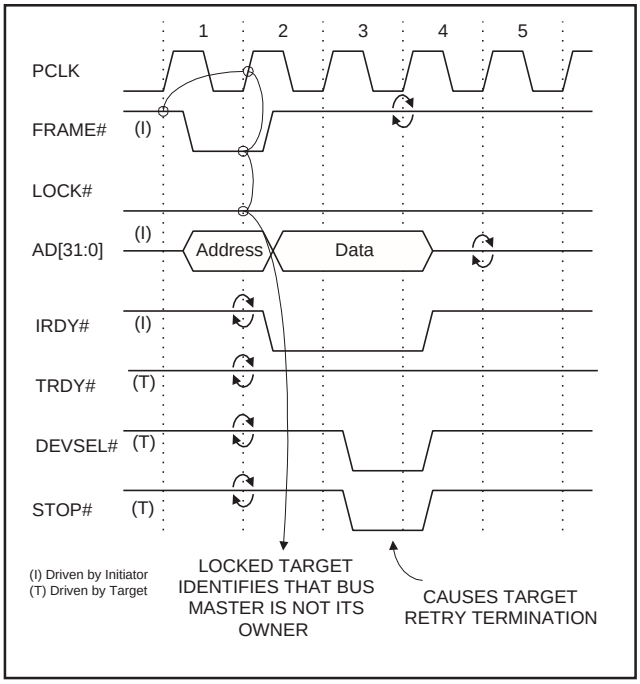
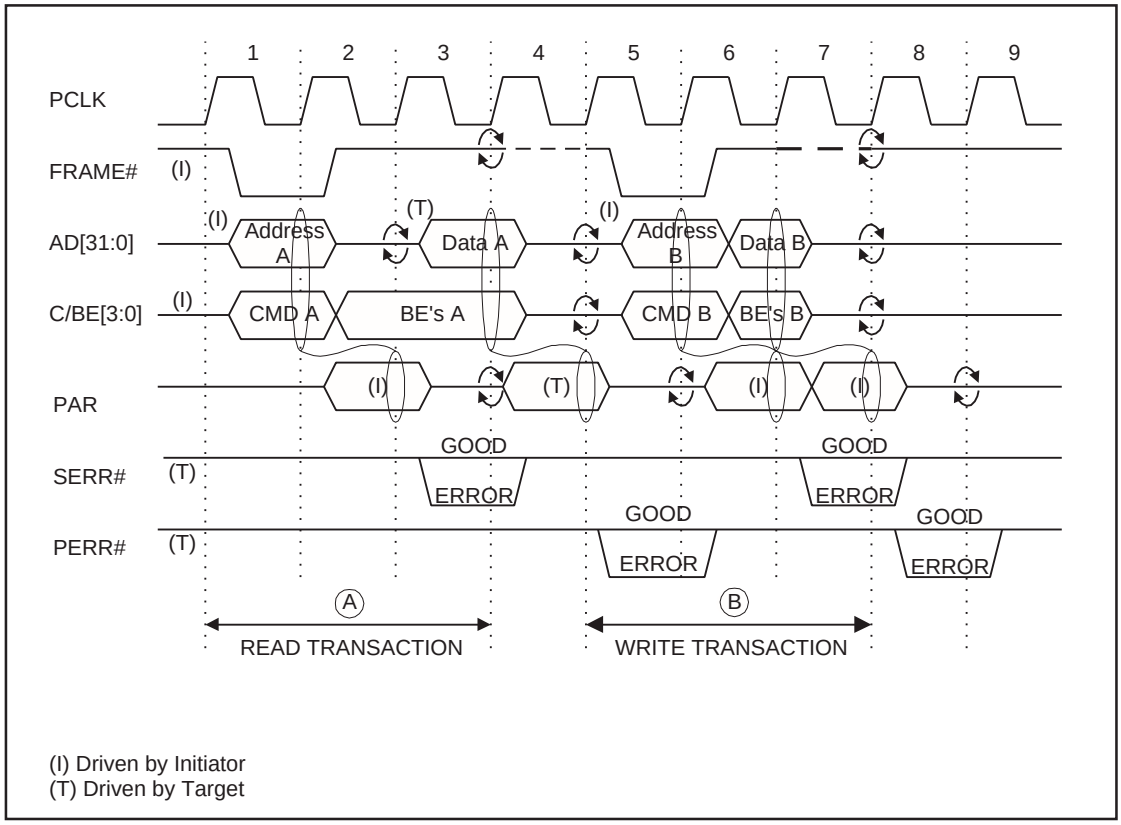


Figure 9. Error Reporting Signal



MAILBOX OVERVIEW

The S5920 has two 32-bit mailbox registers. These mailboxes are useful for passing command and status information between the Add-On and the PCI bus. The PCI interface has one incoming mailbox (Add-On to PCI) and one outgoing mailbox (PCI to Add-On). The Add-On interface has one incoming mailbox (PCI to Add-On) and one outgoing mailbox (Add-On to PCI). The PCI incoming and Add-On outgoing mailboxes are the same, internally. The Add-On incoming and PCI outgoing mailboxes are also the same, internally.

The mailbox status may be monitored in two ways. The PCI and Add-On interfaces each have a mailbox status register to indicate the empty/full status of data bytes within the mailboxes. The mailboxes may also be configured to generate interrupts to the PCI and/or Add-On interface. The outgoing and the incoming mailbox on each interface can be configured to generate interrupts.

FUNCTIONAL DESCRIPTION

Figure 1 shows a block diagram of the PCI to Add-On mailbox registers. Add-On incoming mailbox read accesses pass through an output interlock register. This prevents a PCI bus write to a PCI outgoing mailbox from corrupting data being read by the Add-On. Figure 2 shows a block diagram of the Add-On to PCI mailbox registers. PCI incoming mailbox reads also pass through an interlocking mechanism. This prevents an Add-On write to an outgoing mailbox from corrupting data being read by the PCI bus. The following sections describe the mailbox flag functionality and the mailbox interrupt capabilities.

Figure 1. PCI to Add-On Mailbox Register

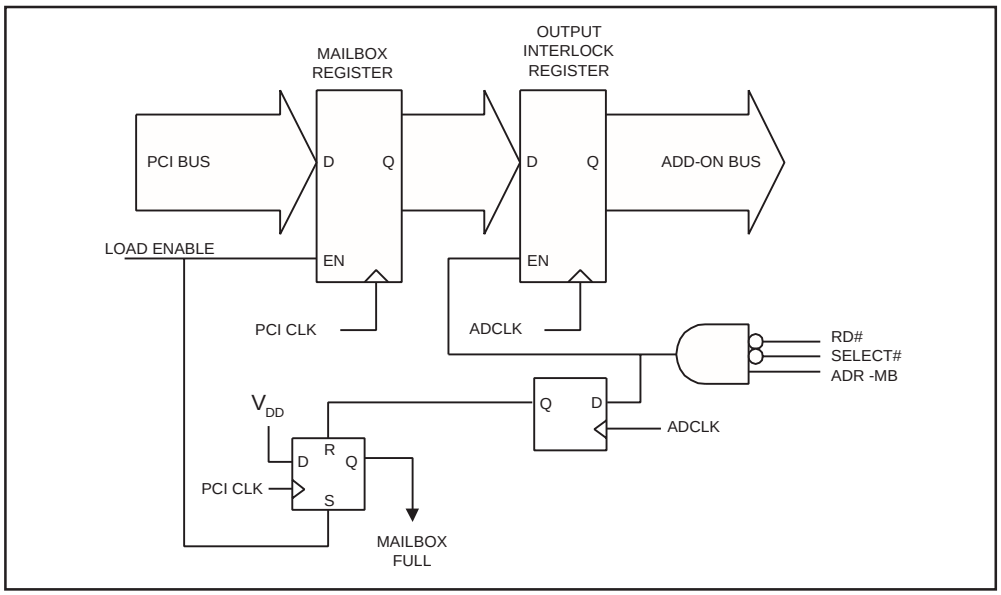
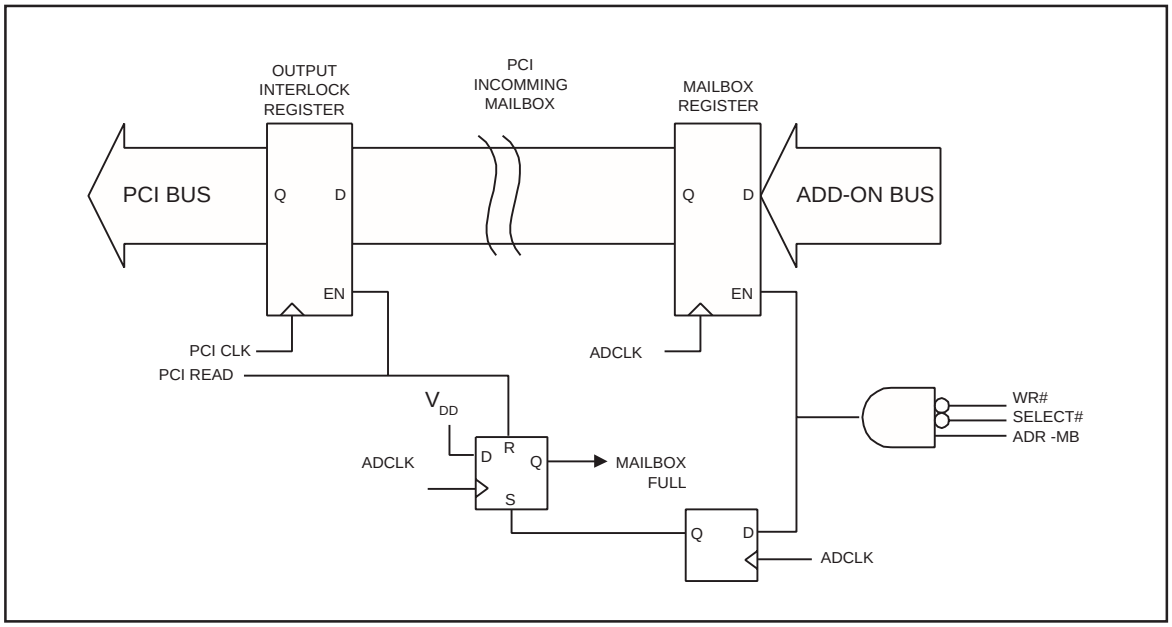


Figure 2. Add-On to PCI Mailbox Register



Mailbox Empty/Full Conditions

The PCI and Add-On interfaces each have a mailbox status register. The PCI Mailbox Empty/Full Status (MBEF) and Add-On Mailbox Empty/Full Status (AMBEF) registers indicate the status of all bytes within the mailbox registers. A write to an outgoing mailbox sets the status bits for that mailbox. The byte enables determine which bytes within the mailbox become full (and which status bits are set).

An outgoing mailbox for one interface is an incoming mailbox for the other. Therefore, incoming mailbox status bits on one interface are identical to the corresponding outgoing mailbox status bits on the other interface. The following list shows the relationship between the mailbox registers on the PCI and Add-On interfaces.

PCI Interface	**Add-On Interface**
Outgoing Mailbox=	Incoming Mailbox
Incoming Mailbox=	Outgoing Mailbox
PCI Mailbox Empty/Full=	Add-On Mailbox Empty/Full

A write to an outgoing mailbox also writes data into the incoming mailbox on the other interface. It also sets the status bits for the outgoing mailbox and the status bits for the incoming mailbox on the other interface. Reading the incoming mailbox clears the corresponding status bit(s) in the Add-On and PCI mailbox status registers (AMBEF and MBEF).

For example, a PCI write is performed to the PCI outgoing mailbox, writing bytes 0 and 1 (CBE0# and CBE1# asserted). Reading the PCI Mailbox Empty/Full Status Register (MBEF) indicates that bits 12 and 13 are set. These bits indicate that outgoing mailbox bytes 0 and 1 are full. Reading the Add-On Mailbox Empty/Full Status Register (AMBEF) shows that bits 12 and 13 in this register are also set, indicating the Add-On incoming mailbox bytes 0 and 1 are full. An Add-On read of the incoming mailbox, bytes 0 and 1, clears the status bits in both the MBEF and AMBEF status registers.

The read-only status flags in the MBEF and AMBEF registers are reset when the corresponding byte is read from the incoming mailbox. Alternately, these flags can be globally reset from either the PCI interface or the Add-On interface. The PCI Bus Reset Control Register (RCR) and the Add-On Reset Control Register (ARCR) each have a bit to reset all of the mailbox status flags.

Mailbox Interrupts

The designer has the option to generate interrupts to the PCI and Add-On interfaces when specific mailbox events occur. The PCI and Add-On interfaces can each define two conditions where interrupts may be generated. An interrupt can be generated when the incoming mailbox becomes full and/or when the outgoing mailbox becomes empty. A specific byte within a specific mailbox is selected to generate the interrupt. The conditions defined to generate interrupts to the PCI interface do not have to be the same as the conditions defined for the Add-On interface.

For the incoming mailbox interrupts, when the specified byte becomes full, an interrupt is generated. The interrupt might be used to indicate command or status information has been provided, and must be read. For PCI incoming mailbox interrupts, the S5920 asserts the PCI interrupt, INTA#. For Add-On incoming mailbox interrupts, the S5920 asserts the Add-On interrupt, IRQ#.

For the outgoing mailbox interrupts, when the specified byte becomes empty, an interrupt is generated. The interrupt might be used to indicate that the other interface has received the last information sent and more may be written. For PCI outgoing mailbox interrupts, the S5920 asserts the PCI interrupt, INTA#. For Add-On outgoing mailbox interrupts, the S5920 asserts the Add-On interrupt, IRQ#.

Add-On Outgoing Mailbox, Byte 3 Access

PCI incoming mailbox byte 3 (Add-On outgoing mailbox, byte 3, or AOMB[3]) has been further enhanced by the addition of a separate 8-bit interface (MD[7:0]) on the Add-On side. This interface can be used to write to AOMB[3] instead of/or in addition to the normal method (via an Add-On Operation Register write to AOMB[3]). The MD[7:0] bus can be configured in one of two modes: Input mode or I/O mode. If the configuration pin MDMODE is strapped high, the MD[7:0] bus is set to be in input mode only. If MDMODE is strapped low, the MD[7:0] bus will operate in a bi-directional mode. If the MD[7:0] bus is set up for input-only mode, data will be latched into AOMB[3] when the LOAD# input is sampled low by the Add-On clock. The LOAD# input pin may also be used to generate a PCI interrupt if the appropriate interrupt is enabled in the Interrupt Control/Status Register (INTCSR). These functions are identical to the Add-On device writing to its byte 3 outgoing mailbox via the DQ bus. As a matter of fact, Add-On mailbox byte 3 is accessible by either the external mailbox port *or* the Add-On interface. Whichever interface writes to it last will determine the data that resides in that register.

Signal Pin	Add-On Outgoing Mailbox
MD0	Mailbox,Bit 24
MD1	Mailbox, Bit 25
MD2	Mailbox, Bit 26
MD3	Mailbox, Bit 27
MD4	Mailbox, Bit 28
MD5	Mailbox,Bit 29
MD6	Mailbox, Bit 30
MD7	Mailbox, Bit 31

When the MD[7:0] bus is set up for I/O mode and LOAD# is high (deasserted), the MD[7:0] bus is an active output, driving the contents of the PCI outgoing mailbox, byte 3 (OMB[3]). In this case, the MD[7:0] bus will be updated anytime the PCI writes to mailbox OMB[3]. As a result, the MD[7:0] bus will be synchronous to the PCI clock. When LOAD# is driven low, the MD[7:0] bus is tri-stated, allowing external data to be latched into Add-On outgoing mailbox byte 3. This is a similar function that exists for input-only.

Figures 3 and 4 show the interaction between the MD[7:0] bus and the LOAD# input pin. Note that a turnaround cycle is utilized when writing data to the mailbox byte in I/O mode. This is to prevent contention on the MD[7:0] drivers.

BUS INTERFACE

The mailboxes appear on the Add-On and PCI bus interfaces as two operation registers. One is the outgoing mailbox, and the other is the incoming mailbox. These mailboxes may be used to generate interrupts to each of the interfaces. The following sections describe the Add-On and PCI bus interfaces for the mailbox registers.

PCI Bus Interface

The mailbox operation registers do not support burst accesses by an initiator. A PCI initiator attempting to burst to the mailbox registers causes the S5920 to respond with a target disconnect with data. PCI writes to a full outgoing mailbox overwrite data currently in that mailbox. PCI reads from an empty incoming mailbox return the data that was previously contained in the mailbox. In this case, the data cannot be guaranteed. It is intended for the user to verify that a mailbox is full before it is read.

PCI incoming and outgoing mailbox interrupts are enabled/disabled in the INTCSR. The mailboxes can generate a PCI interrupt (INTA#) under two conditions (individually enabled). For an incoming mailbox full interrupt, INTA# is asserted on the rising edge of the PCI clock after the Add-On mailbox write completes. For an outgoing mailbox empty interrupt, INTA# is asserted on the rising edge of the PCI clock after the Add-On mailbox read completes. INTA# is deasserted one PCI clock cycle after the mailbox interrupt is serviced (by writing a 1 to the proper interrupt source bit).

Add-On Bus Interface

The Add-On mailbox interface behaves similarly to the PCI bus interface. Add-On writes to a full outgoing mailbox overwrite data currently in that mailbox. PCI reads from an empty incoming mailbox return the data that was previously contained in the mailbox.

Add-On incoming and outgoing mailbox interrupts are enabled/disabled in the Add-On Interrupt Control/Status Register (AINT). The mailboxes can generate the Add-On IRQ# interrupt under two conditions (individually enabled). For an incoming mailbox full interrupt, IRQ# is asserted on the rising edge of the Add-On clock after the PCI mailbox write completes. For an outgoing mailbox empty interrupt, IRQ# is asserted on the rising edge of the Add-On clock after the PCI mailbox read completes. IRQ# is deasserted one Add-On clock cycle after the mailbox interrupt is serviced (by writing a 1 to the proper interrupt source bit).

8-Bit and 16-Bit Add-On Interfaces

Some Add-On designs may implement an 8-bit or 16-bit bus interface. The mailboxes do not require a 32-bit Add-On interface for all the bytes to be read/written. For 8-bit interfaces, the 8-bit data bus may be externally connected to all 4 bytes of the 32-bit Add-On interface (DQ 31:24, 23:16, 15:8, 7:0 are all connected). The Add-On device reading or writing the mailbox registers may access all mailbox bytes by cycling through the Add-On byte enable inputs (only one byte enable may be active at a time). A similar solution applies to 16-bit Add-On buses. This solution works for Add-On designs which always use just one bus width (8-bit or 16-bit).

If the DQMODE pin is high, indicating a 16-bit Add-On interface, the previous solution may be used to implement an 8-bit interface. The only modification needed is that BE3(= ADR1) must be toggled after the first two accesses to steer the S5920 internal data bus to access the upper 16 bits of the mailboxes.

CONFIGURATION

The PCI interface and the Add-On interface each have one incoming mailbox (IMB or AIMB) and one outgoing mailbox (OMB or AOMB) along with a single mailbox status register (MBEF or AMBEF). The outgoing mailbox is read/write, the incoming mailbox and the mailbox status registers are read-only.

The following sections discuss the registers associated with the mailboxes and accesses required for different modes of mailbox operation.

Mailbox Status

Every byte in each mailbox has a status bit in the Mailbox Empty/Full Status Registers (MBEF and AMBEF). Writing a particular byte into the outgoing mailbox sets the corresponding status bit in both the MBEF and AMBEF registers. A read of a 'full' byte in a mailbox clears the status bit. The MBEF and AMBEF are read-only. Status bits cannot be cleared by writes to the status registers.

The S5920 allows the mailbox status bits to be reset through software. The PCI Bus Reset Control PCI Operation Register (RCR) and the Add-On Reset Control Add-On Operation Register (ARCR) each have a bit to reset mailbox status. Writing a 1 to Mailbox Flag Reset bit in the RCR or the ARCR register immediately clears all bits in the both the MBEF and AMBEF registers. Writing a 0 has no effect. The Mailbox Flag Reset bit is write-only.

The flag bits should be monitored when transferring data through the mailboxes. Checking the mailbox status before performing an operation prevents data from being lost or corrupted. The following sequences are suggested for PCI mailbox operations using status polling (interrupts disabled).

Figure 3. Input/Output Mode (MDMODE=0)

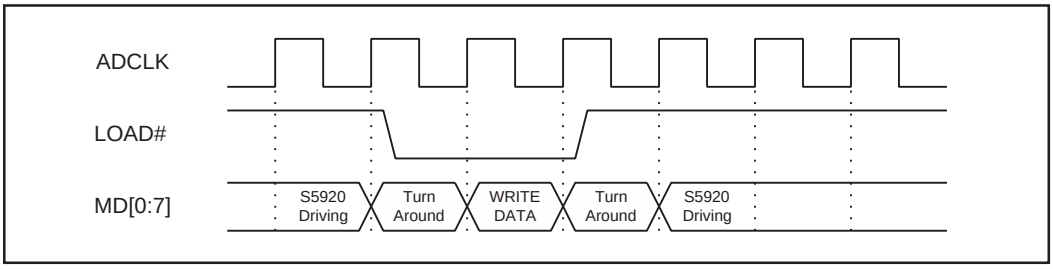
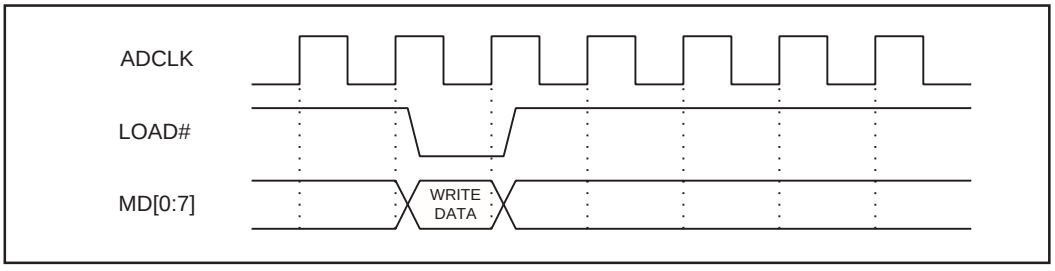


Figure 4. Input Mode (MDMODE=1)



Reading the PCI Incoming Mailbox:

1) Check Mailbox Status. Read the mailbox status register to determine if any information has been passed from the Add-On interface.

   MBEF Bits 31:28 If a bit is set, valid data is contained in the corresponding mailbox byte.

2) Read Mailbox. Read the mailbox bytes which MBEF indicates are full. This automatically resets the status bits in the MBEF and AMBEF registers.

   IMB Bits 31:0 Mailbox data.

Writing the PCI Outgoing Mailbox:

1) Check Mailbox Status. Read the mailbox status register to determine if information previously written to the mailbox has been read by the Add-On interface. Writes to full mailbox bytes overwrite data currently in the mailbox (if not already read by the Add-On interface). Repeat until the byte(s) to be written are empty.

   MBEF Bits 15:12 If a bit is set, valid data is contained in the corresponding mailbox byte and has not been read by the Add-On.

2) Write Mailbox. Write to the outgoing mailbox byte(s).

   OMB Bits 31:0 Mailbox data.

Mailbox operations for the Add-On interface are functionally identical. The following sequences are suggested for Add-On mailbox operations using status polling (interrupts disabled):

Reading an Add-On Incoming Mailbox:

1) Check Mailbox Status. Read the mailbox status register to determine if any information has been passed from the PCI interface.

   AMBEF Bits 15:12 If a bit is set, valid data is contained in the corresponding mailbox byte.

2) Read Mailbox. Read the mailbox bytes which AMBEF indicates are full. This automatically resets the status bits in the AMBEF and MBEF registers.

   AIMB Bits 31:0 Mailbox data.

Writing an Add-On Outgoing Mailbox:

1) Check Mailbox Status. Read the mailbox status register to determine if information previously written to the mailbox has been read by the PCI interface. Writes to full mailbox bytes overwrite data currently in the mailbox (if not already read by the PCI interface). Repeat until the byte(s) to be written are empty.

   AMBEF Bits 31:28 If a bit is set, valid data is contained in corresponding mailbox byte and has not been read by the PCI bus.

2) Write Mailbox. Write to the outgoing mailbox byte(s).

   AOMB Bits 31:0 Mailbox data.

Mailbox Interrupts

Although polling status is useful in some cases, polling requires continuous actions by the processor. Mailbox interrupt capabilities are provided to avoid much of the processor overhead required by continuously polling status bits.

The Add-On and PCI interface can each generate interrupts on the incoming mailbox condition and/or the outgoing mailbox condition. These can be individual enabled/disabled. A specific byte in the incoming mailbox and outgoing mailbox is identified to generate the interrupt(s). The tasks required to setup the mailbox interrupts are as follows:

Enabling PCI mailbox interrupts:

1) Enable PCI outgoing mailbox interrupts. A specific byte within the outgoing mailboxes is identified to assert INTA# when read by the Add-On interface.

INTCSR	Bit 4	Enable outgoing mailbox interrupts
INTCSR	Bits 1:0	Identify mailbox byte to generate interrupt

2) Enable PCI incoming mailbox interrupts. A specific byte within the incoming mailboxes is identified to assert INTA# when written by the Add-On interface.

INTCSR	Bit 12	Enable incoming mailbox interrupts
INTCSR	Bits 9:8	Identify mailbox byte to generate interrupt

Enabling Add-On mailbox interrupts:

1) Enable Add-On outgoing mailbox interrupts. A specific byte within the outgoing mailboxes is identified to assert IRQ# when read by the PCI interface.

AINT	Bit 12	Enable outgoing mailbox interrupts
AINT	Bits 9:8	Identify mailbox byte to generate interrupt

2) Enable Add-On incoming mailbox interrupts. A specific byte within the incoming mailboxes is identified to assert IRQ# when written by the PCI interface.

AINT	Bit 4	Enable incoming mailbox interrupts
AINT	Bits 1:0	Identify mailbox byte to generate interrupt

With either the Add-On or PCI interface, these two steps can be performed with a single access to the appropriate register. They are shown separately here for clarity.

Once interrupts are enabled, the interrupt service routine must access the mailboxes and clear the interrupt source. A particular application may not require all of the steps shown. For instance, a design may only use the incoming mailbox interrupts and not require support for the outgoing mailbox interrupts. The interrupt service routine tasks are as follows:

Servicing a PCI Mailbox Interrupt (INTA# asserted):

1) Identify the interrupt source(s). Multiple interrupt sources are available on the S5920. The interrupt service routine must verify that a mailbox generated the interrupt (and not some other interrupt source).

INTCSR	Bit 23	PCI interrupt asserted
INTCSR	Bit 17	PCI incoming mailbox interrupt indicator
INTCSR	Bit 16	PCI outgoing mailbox interrupt indicator

2) Check mailbox status. The mailbox status bits indicate which mailbox bytes must be read or written.

MBEF	Bits 31:28	Full PCI incoming mailbox bytes
MBEF	Bits 15:12	Empty PCI outgoing mailbox bytes

3) Access the mailbox. Based on the contents of MBEF, mailboxes are read or written. Reading an incoming mailbox byte clears the corresponding status bit in MBEF.

OMB	Bits 31:0	PCI outgoing mailboxes
IMB	Bits 31:0	PCI incoming mailboxes

4) Clear the interrupt source. The PCI INTA# signal is deasserted by clearing the interrupt request. The request is cleared by writing a 1 to the appropriate bit.

INTCSR	Bit 17	Clear PCI incoming mailbox interrupt
INTCSR	Bit 16	Clear PCI outgoing mailbox interrupt

Servicing the Add-On mailbox interrupt (IRQ# asserted):

1) Identify the interrupt source(s). Multiple interrupt sources are available on the S5920. The interrupt service routine must verify that a mailbox generated the interrupt (and not some other interrupt source).

AINT	Bit 23	Add-On interrupt asserted
AINT	Bit 17	Add-On outgoing mailbox interrupt indicator
AINT	Bit 16	Add-On incoming mailbox interrupt indicator

2) Check mailbox status. The mailbox status bits indicate which mailbox bytes must be read or written.

AMBEF	Bits 31:28	Empty Add-On outgoing mailbox bytes
AMBEF	Bits 15:12	Full Add-On incoming mailbox bytes

3) Access the mailbox. Based on the contents of AMBEF, mailboxes are read or written. Reading the incoming mailbox byte clears the corresponding status bit in AMBEF.

AIMB	Bits 31:0	Add-On incoming mailbox
AOMB	Bits 31:0	Add-On outgoing mailbox

4) Clear the interrupt source. The Add-On IRQ# signal is deasserted by clearing the interrupt request. The request is cleared by writing a 1 to the appropriate bit.

AINT	Bit 17	Clear Add-On outgoing mailbox interrupt
AINT	Bit 16	Clear Add-On incoming mailbox interrupt

NOTE: For an incoming mailbox interrupt, step 3 involves accessing the mailbox. To allow the incoming mailbox interrupt logic to be cleared, the mailbox status bit must also be cleared. Reading an incoming mailbox clears the status bits. Another option for clearing the status bits is to use the Mailbox Flag Reset bit in the RCR and ARCR registers, but this clears all status bits, not just a single mailbox byte. For outgoing mailbox interrupts, the status bit was already cleared prior to the generation of the interrupt. As a result, the mailbox does not need to be read.

ADD-ON LOCAL BUS INTERFACE

This chapter describes the Add-On Local bus interface of the S5920. The S5920 is designed to support connection to a variety of microprocessor buses and/or peripheral devices. The Add-On interface controls S5920 operation through the Add-On Operation Registers accessed through the 32 bit local bus.

The Add-On local bus interface is synchronous to ADCLK. ADCLK is a 0-40 MHz clock input which can be configured as asynchronous to the PCI clock or synchronous when connected to the S5920 BPCLK output. The following sections describe the various interfaces to the PCI bus and how they are accessed from the Add-On bus.

ADD-ON INTERFACE SIGNALS

The Add-On bus provides a number of system signals to allow Add-On logic to monitor PCI bus activity, to indicate status conditions (interrupts), and to configure the S5920 Add-On bus.

SYSTEM SIGNALS

BPCLK is a buffered version of the PCI clock. The PCI clock can operate from 0 MHz to 33 MHz.

SYSRST# is a buffered version of the PCI reset signal, and may also be toggled by host application software through bit 24 of the Reset Control Register (RCR).

IRQ# is the PCI interrupt request output to the Add-On bus. This signal is active low and can indicate multiple conditions. Add-On interrupts can be generated from the mailbox interface or to indicate start of BIST. The conditions which will generate an IRQ# due to mailbox activity are discussed in the mailbox chapter. The IRQ# output is deasserted when acknowledged by writing a 1 to the corresponding interrupt bit in the Add-On Interrupt Control/Status Register (AINT). See Table 3.

The PTMODE signal (Pass-Thru Mode) controls the Pass-Thru interface only. Asserting it will configure the Pass-Thru in Passive mode and low will configure the Pass-Thru in Active mode.

ADDINT# is an Add-On interrupt input pin. When asserted, it will cause the PCI interrupt output pin (INTA#) to assert. The ADDINT# is a level-sensitive input. Any number of Add-On peripheral interrupt sources can drive this input. There must be a pull-up resistor on the board to pull it high when inactive. This interrupt has to be enabled by Bit 13 of the INTCSR. It is the responsibility of the PCI host to clear the interrupt source of ADDINT# in order to have the pending interrupt deasserted.

The DQMODE signal configures the data path width for all Add-On Operation register accesses, except for the Pass-Thru Data and Address registers. When DQMODE is low, DQ is configured as a 32-bit data bus. When DQMODE is high, DQ is configured as a 16-bit data bus. For 16-bit operation, BE3# is redefined as ADR1, providing an extra address input, and BE2# is unused. ADR1 selects the low or high words of the 32-bit S5920 Add-On Operation Registers.

ADD-ON S5920 REGISTER ACCESSES

The S5920 Add-On bus is very similar to that of a memory or peripheral device found in a microprocessor-based system. A 32-bit data bus with individual read and write strobes, a chip select and byte enables are provided.

Register Access Signals

Register accesses to the S5920 Add-On Operation Registers are synchronous to the Add-On input clock (ADCLK). The following signals are required to complete a register access to the S5920.

BE[3:0]# Byte Enable Inputs. These signals identify which bytes of the DQ bus are valid during Add-On bus transactions. BE0# indicates valid DQ[7:0], BE1# a valid DQ[15:8], etc. When DQ is configured for 16-bit operation, BE2# is not defined and BE3# becomes ADR1.

ADR[6:2] Address Register Inputs. These pins address a specific Add-On Operation Register within the S5920. When DQ is configured for 16-bit operation, an additional input, ADR1 is available to allow the 32-bit operation registers to be accessed in two 16-bit cycles.

RD# Read Enable Input.

WR# Write Enable Input.

SELECT# Chip Select Input. This input indicates RD#, WR#, ADR[6:2] and BE[3:0] are valid.

DQ[31:0] Bi-directional Data Bus. These I/O pins are the Add-On data bus.

S5920 General Register Accesses

For many Add-On applications, Add-On logic does not operate at the PCI bus frequency. This is especially true for Add-On designs implementing a microprocessor, which may be operating at a lower or higher frequency.

The RD# and WR# inputs become enables, using ADCLK to clock data into and out of registers. All inputs are sampled on the rising edge of ADCLK.

Figures 1 and 2 show basic operation register access timing relationships. Detailed AC timings are in Electrical and AC Characteristics. Chapter 10.

For reads (Figure 1), data is driven onto the DQ bus on the ADCLK cycle after RD# is sampled asserted. When RD# is not asserted, the DQ outputs float. The address, byte enable, and RD# inputs must meet setup and hold times relative to the rising edge of ADCLK.

For writes (Figure 2), data is clocked into an operation register on the rising edge of ADCLK in which WR# is sampled asserted. Address, byte enables, WR# and data must all meet setup and hold times relative to the rising edge or ADCLK.

Figure 1. Read Operation Register

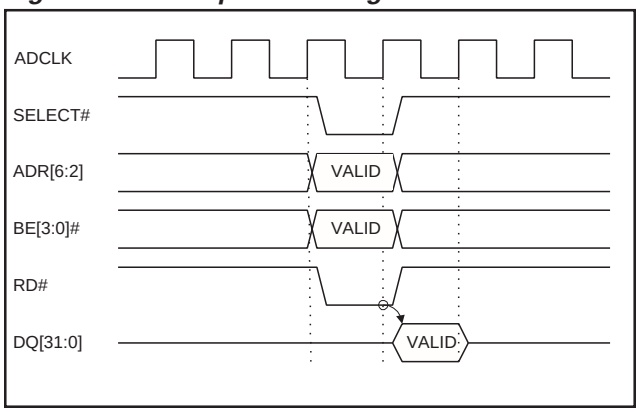
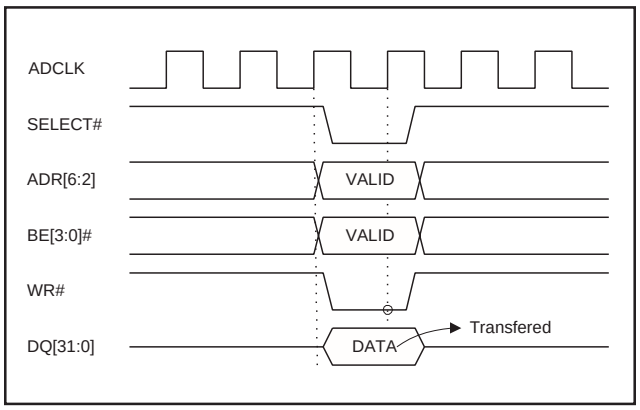


Figure 2. Write Operation Register



S5920 16-bit Mode Register Accesses

In the S5920 there are two methods of defining the Add-On DQ width: one is by using the DQMODE pin and the other is to define a Pass-Thru region size of 8, 16 or 32 bits. The DQMODE pin allows external 16-bit devices to access S5920 Operation Registers without additional logic. The external device is able to write and read the S5920 32-bit registers in two 16-bit cycle accesses. When performing an Operation Register access with the DQMODE pin set for 16 bits (DQMODE = 1), only the lower half (DQ[15:0]) of the DQ bus is driven during a read or write. The S5920 internally steers the data bus and the byte enables based on the BE3# input. It is important to note that the DQMODE pin has *no* effect on accesses to the Pass-Thru Data Register. For non 32-bit Pass-Thru regions, the region size should be used instead. In 16-bit mode, a 32-bit DWORD write is performed in two cycles:

Cycle 1: DQ[15:0] is driven with the lower-WORD. WR#, ADR and SELECT# are asserted, BE[1:0]# indicates which bytes of the WORD are valid, and BE3# (which has been redefined as ADR1 when in 16-bit mode) is set to zero, indicating that the write is for the lower-WORD of the DWORD transfer. DQ[15:0] will be written to the bottom 16 bits of the internal 32-bit register (be it a Mailbox, Pass-Thru configuration register, etc.).

Cycle 2: DQ[15:0] is driven with the upper-WORD. WR#, ADR and SELECT# are asserted, BE[1:0]# indicate which bytes of the WORD are valid, and BE3# is set to one, indicating that the write is for the upper-WORD of the DWORD transfer. DQ[15:0] will be written to the upper 16-bits of the internal 32-bit register.

Figure 3. 16 Bit Mode Operation Register DWORD Write/Read

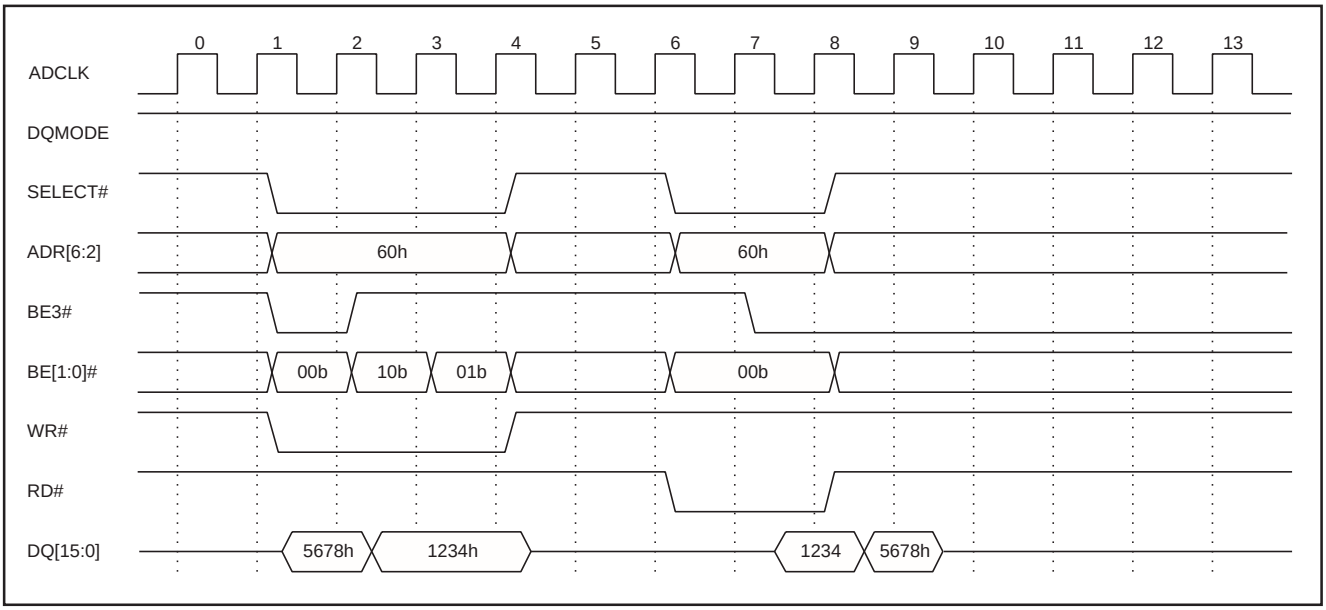


Figure 3 shows an example of a DWORD write of 12345678h using a 16 bit-mode write transfer, and is described as follows:

Clock 1: ADR[6:2], BE[3:0], SELECT# and WR# are driven, DQ[15:0] is driven with the data to be written. BE3# is low indicating that the DQ bus data is to be written to the lower WORD of the register. BE[2:0]# is 00h, indicating that both bytes on the DQ[15:0] bus are valid and should be written to the register indicated by ADR[6:2].

Clock 2: The rising edge of clock 2 writes 5678h into the lower WORD of the register. 1234h is driven onto the DQ[15:0] bus. BE3# is driven high, indicating the DQ bus data is to be written to the upper WORD of the register. BE[2:0]# is 10h indicating that the lower byte of the WORD on DQ[15:0] bus is valid. This example shows how the BEs function.

Clock 3: The rising edge of clock 3 writes 34h into the lower byte of the upper WORD of the register. BE[2:0]# is “01” indicating the upper byte on DQ[15:0] is valid.

Clock 4: The rising edge of clock 4 writes 12h into the upper byte of the upper WORD of the register. 12345678h is in the register selected by ADR[6:2]. SELECT#, ADR[6:2], WR#, BE[3:0]# and DQ are deasserted. No read or write occurs on the rising edge of clocks 5 and 6.

Figure 3 also shows a DWORD read of 12345678h from the same register, using a 16-bit mode read transfer, and is described as follows:

Clock 6: ADR[6:2], SELECT# and RD# are asserted. BE3# is high, indicating the upper WORD of the register is to be driven onto DQ[15:0] and BE[1:0]# is 00h indicating both bytes of the WORD are to be driven.

Clock 7: The S5920 drives 1234h onto DQ[15:0] as a result of the read issued during the previous cycle. BE3# is next driven low to indicate the lower WORD of the register is to be driven onto DQ[15:0]. BE[1:0]# is 00h indicating both bytes of the WORD should be driven onto DQ[15:0]. Note: in the event that BE[0]# was 1b, DQ[7:0] would NOT be driven during Clock 8, it would remain tri-state. The only exception to this is if ADR[6:2] indicated the Pass-Thru Data Register, where all of DQ[15:0] would be driven, regardless of the state of BE[1:0]#.

Clock 8: On the rising-edge, Add-On logic latches data 1234h. The S5920 drives 5678h onto DQ[15:0] as a result of the read issued during the previous cycle. ADR[6:2], SELECT#, RD# and BE[3:0]# are deasserted, completing the transfer.

Clock 9: On the rising-edge, Add-On logic latches data 5678h. DQ[15:0] returns to tri-state as RD# was sampled deasserted.

MAILBOX OVERVIEW

For a detailed description of the Mailbox interface, reference Chapter 8.

PASS-THRU OVERVIEW

The S5920 provides data transfers between the PCI bus and the user local bus through the Pass-Thru data channel. Using a handshaking protocol with Add-On device(s), the PCI bus can directly access data on the Add-On bus and internal S5920 Operation registers. The Pass-Thru data channel is very flexible for user memory access or accessing registers within peripherals on the Add-On bus. Pass-Thru operation in

Active or Passive mode requires an external non-volatile memory device to define and configure the Pass-Thru channel region sizes and bus widths.

Four user-configurable Pass-Thru regions are available in the S5920. Each region is defined by a PCI Configuration Base Address Register (BADR1-4). A Pass-Thru region defines a block of pre-defined user space address in either host memory or I/O areas. Memory mapped regions can be requested below 1 Mbyte (Real Mode address space for a PC). Each region is configurable for bus widths of 8, 16 or 32 bits for the Add-On bus interface.

The S5920 Pass-Thru channel supports single data transfers as well as burst transfers. When accessed with burst transfers, the S5920 supports data transfers at the full PCI bandwidth. The data transfer rate is only limited by the PCI initiator performing the access and the speed of the Add-On bus logic.

WRITE FIFO OVERVIEW

For PCI write cycles, the S5920 has an 8x32-bit Write FIFO to increase performance for slow Add-On devices. When the FIFO is enabled, the S5920 will accept data transfers from the PCI bus at zero wait states until the FIFO is full. The device continues to fill the FIFO as long as the transfers are sequential. The S5920 can continue accepting sequential write PCI transfers as long as the FIFO is not full and the boundary of the Pass-Thru region defined by the Base Address Register is not crossed. If the next data access is for a non-sequential address, the FIFO must first be emptied by the Add-On peripheral in order for the next transfer to occur.

The Write FIFO can be disabled, thus configuring the FIFO to act as a single DWORD data buffer. In this case, PCI Write Posting is not possible.

READ FIFO OVERVIEW

The S5920 has an 8x32-bit Read FIFO, which allows data to be prefetched from the add on bus. The user can program the device to prefetch 2,4 or 8 DWORDs for each region or disable prefetching completely. For the first PCI read cycle, the device will request data from the Add-On bus. As the PCI bus reads the FIFO, and until after the PCI transfer has finished, the S5920 will prefetch the next N (2, 4 or 8) DWORDs from the Add-On. The prefetched data is valid as long as the PCI read addresses are sequential. If the current PCI read address is not the previous address plus four, or if a PCI write access occurs, the S5920 will flush the FIFO and start a new transfer at this address. Flushing the FIFO will incur a minimum loss of one PCI clock cycle or possibly more if the Add-On logic has not finished its current prefetching transfer. Note that prefetching is not performed past the upper limit of the base address region. In fact, prefetching is disabled when the PCI address is eight DWORDs from the end of the region. Prefetch cycles are always 32 bits regardless of Add-On bus width or the byte enables requested by the PCI.

FUNCTIONAL DESCRIPTION

The S5920 Pass-Thru interface supports both single cycle (one data phase) and burst accesses (multiple data phases).

Pass-Thru Transfers

The Pass-Thru interface offers two different modes of operation: Passive mode and Active mode. Passive mode is configured by strapping the pin PTMODE high, while Active mode is configured by strapping the pin PTMODE Low.

PTMODE = 1 - Passive Operation
PTMODE = 0 - Active Operation

Passive operation allows external Add-On bus peripherals to provide read and write control signals to the S5920. The user drives SELECT#, RD#, WR#. ADR[6:2] and PTRDY#. The Add-On bus logic has the flexibility of determining when it wants to perform reads/writes.

Some applications may require that a PCI address be passed for Pass-Thru accesses. For example, a 4-Kbyte Pass-Thru region on the PCI bus may correspond to a 4-Kbyte block of SRAM on the Add-On card. If a PCI initiator accesses this region, the Add-On would need to know the offset within the memory device to access. The Pass-Thru Address Register (APTA) allows Add-On logic to access address information for the current PCI cycle. When the PCI bus performs burst accesses, the APTA register is incremented by the S5920 to reflect the address of the current data phase. PTNUM[1:0] is used to determine what region owns the current data access.

For PCI writes to the Add-On, the S5920 transfers data from the PCI bus into the Pass-Thru Write FIFO. When the Pass-Thru write FIFO becomes not empty, the S5920 asserts the Pass-Thru status signals to indicate to the Add-On that data is present. The Add-On logic will then read data from the FIFO. The S5920 continues accepting write data from the PCI initiator as long as the 8x32 FIFO is not full.

For PCI reads from the Add-On, the S5920 asserts the Pass-Thru status signals to indicate to the Add-On that data is required. The Add-On logic should write the requested data into the Pass-Thru Read FIFO. The S5920 will assert TRDY# to the PCI bus after the Add-On logic has transferred data into the FIFO. As long as data is in the FIFO, and PCI read data is still requested, TRDY# will continue to be asserted. If the Add-On cannot provide data quickly enough, the S5920 signals a disconnect to the PCI bus. This allows the PCI bus to perform other tasks, rather than waiting for a slow target. The S5920 will prefetch data if enabled.

Signal	Function
PTATN#	This output indicates a Pass-Thru access needs servicing.
PTBURST#	This output indicates that the current Pass-Thru access is a PCI burst transfer or a single cycle transfer. PTBURST# is deasserted immediately after the second to last burst data has been transferred. PTBURST# is also active during prefetch cycles.
PTNUM[1:0]	These outputs indicate which Pass-Thru region decoded the PCI address.
PTBE[3:0]#	These outputs indicate which data bytes are valid (PCI writes), or requested (PCI reads). See timing diagrams for further details. PTBE0# = 0 Byte 0 is valid, PTBE1# = 0 Byte 1 is valid, PTBE2# = 0 Byte 2 is valid, PTBE3# = 0 Byte 3 is valid
PTWR	This output indicates if the Pass-Thru access is a PCI read or a write.
PTADR#	When asserted, this pin drives the Pass-Thru Address Register contents onto the Add-On data bus. This input enables the DQ[31:0] data bus to become active immediately. There is NO pipeline delay from PTADR# to DQ, as there is from RD# to DQ. As a result, this is an asynchronous input.
PTRDY#	In Passive mode, this input indicates the current Pass-Thru transfer has been completed by the Add-On. In Active mode, this input indicates that wait states are to be inserted for the next transfer.
ADCLK	Input Add-On clock (to synchronize Pass-Thru data register accesses)

Pass-Thru Status/Control Signals

The S5920 Pass-Thru registers are accessed using the Add-On register access pins. The Pass-Thru Address Register (APTA) can, optionally, be accessed using a single, direct access input, PTADR#. Pass-Thru cycle status indicators are provided to control Add-On logic based on the type of Pass-Thru access occurring (single cycle, burst, etc.). The signals in the table above are provided for Pass-Thru operation:

BUS INTERFACE

The Pass-Thru data channel allows PCI initiators to read or write to resources on the Add-On bus. A PCI initiator may access the Add-On with single data phase cycles or multiple data phase bursts. The Add-On interface implements Pass-Thru status and control signals used by logic to complete data transfers initiated by the PCI bus. The Pass-Thru interface is designed to allow Add-On logic to function without knowledge of PCI bus activity. Add-On logic only needs to react to the Pass-Thru status signals. The S5920 PCI device independently interacts with the PCI initiator to control data flow between the devices.

The following sections describe the PCI and Add-On bus interfaces. The PCI interface description provides a basic overview of how the S5920 interacts with the PCI bus, and may be useful in system debugging. The Add-On interface description indicates functions required by Add-On logic and details the Pass-Thru handshaking protocol.

PCI Bus Interface

The S5920 device examines all PCI bus cycle addresses. If the address associated with the current cycle decodes to one of the S5920 Pass-Thru regions, DEVSEL# is asserted. If the Pass-Thru logic is currently idle (not busy finishing a previous Pass-Thru operation), the bus cycle type is decoded and the Add-On Pass-Thru status outputs are set to initiate a transfer on the Add-On bus.

The following sections describe the behavior of the PCI interface for Pass-Thru accesses to the S5920. Single cycle accesses, burst accesses, and target-initiated retries are detailed.

PCI Pass-Thru Single Cycle Accesses

A single cycle transfer is the simplest of PCI bus transactions. Single cycle transfers have an address phase and a single data phase. The PCI bus transaction starts when an initiator drives address and command information onto the PCI bus and asserts FRAME#. The initiator always deasserts FRAME# before the last data phase. For single cycle transfers, FRAME# is only asserted during the address phase (indicating the first data phase is also the last).

When the S5920 sees FRAME# asserted, it samples the address and command information to determine if the bus transaction is intended for it. If the address is within one of the defined Pass-Thru regions or internal

PCI Operation Register, the S5920 accepts the transfer (asserts DEVSEL#), and stores the PCI address in the Pass-Thru Address Register (APTA).

For Pass-Thru writes, the S5920 responds immediately (asserting TRDY#) and transfers the data from the PCI bus into the write FIFO as long as the write FIFO is not full. The S5920 then indicates to the Add-On interface that a Pass-Thru write is taking place and waits for Add-On logic to complete the transfer. Once the S5920 has captured the data from the PCI bus, the transfer is finished from the PCI bus perspective, and the PCI bus becomes available for other transfers.

For Pass-Thru reads, the S5920 indicates to the Add-On interface that a Pass-Thru read is taking place and waits for Add-On logic to complete the cycle. If the Add-On cannot complete the cycle quickly enough, the S5920 requests a retry from the initiator. The S5920 will fetch one DWORD from the Add-On side, and store it in the Read FIFO.

PCI Pass-Thru Burst Accesses

For PCI Pass-Thru burst accesses, the S5920 captures the PCI address and determines if it falls into one of the defined Pass-Thru regions. Accesses that fall into a Pass-Thru region or internal PCI Operation Register are accepted by asserting DEVSEL#. The S5920 monitors FRAME# and IRDY# on the PCI bus to identify burst accesses. If the PCI initiator is performing a burst access, the Pass-Thru status indicators notify the Add-On logic.

For Pass-Thru burst writes, the S5920 responds immediately (asserting TRDY#). The S5920 transfers the first data phase of the burst into the FIFO, and stores the PCI address in the Pass-Thru Address Register (APTA). The S5920 can accept up to 8 DWORDs from the PCI bus before transferring one DWORD on the Add-On side. If the Add-On bus is slow, the device will keep the FIFO full until the data is ready to be transferred by the slow Add-On bus. If the Add-On bus is fast at accepting the data, then the FIFO will continue an indefinite burst, or until the PCI master is forced to relinquish the bus for arbitration reasons, or the PCI bus master has gone beyond the Pass-Thru region address space. For burst accesses, the APTA is automatically incremented by the S5920 for each data phase.

For Pass-Thru burst reads, the S5920 claims the PCI cycle (asserting DEVSEL#). The request for data is passed on to Add-On logic and the PCI address is stored in the APTA register. The device will prefetch data if the feature is enabled. The S5920 then drives the requested data on the PCI bus and asserts TRDY# to begin the next data phase. The APTA register is automatically incremented by the S5920 after each data phase.

PCI Disconnect Conditions

Before discussing what causes the S5920 to issue a disconnect on the PCI bus, it might be useful to distinguish between a disconnect and retry. A retry occurs when a PCI initiator does not receive a single TRDY#, but is issued a STOP# instead. In this case, no data is transferred. The PCI 2.1 spec states that the initiator is required to come back and complete this transfer. A disconnect occurs after at least one data phase was completed (TRDY# and IRDY# asserted simultaneously). This occurs when a STOP# is asserted either with a TRDY# or after a TRDY#/IRDY# transfer. In this case, the initiator is not required to return to complete the transfer.

In some applications, Add-On logic may not be able to respond to Pass-Thru accesses quickly. In this situation, the S5920 will Retry the cycle on the PCI side. For PCI write cycles, the S5920 will accept up to 8 DWORDs without a disconnect or until the FIFO is full. For a PCI read cycle, the first access needs to take less than 16 PCI clocks, otherwise the device will issue a Retry. A subsequent read transfer must take less than 8 PCI clocks, otherwise the device will issue a disconnect.

With many devices, particularly memories, the first access takes longer than subsequent accesses (assuming they are sequential and not random). For this reason, the PCI specification allows 16 clocks to respond to the first data phase of a PCI cycle and 8 clocks for subsequent data phases (in the case of a burst) before a retry/disconnect is issued by the S5920.

The S5920 also requests a disconnect if an initiator attempts to burst past the end of a Pass-Thru region. The S5920 updates the Pass-Thru Address Register (APTA) for each data phase during bursts, and if the updated address is not within the current Pass-Thru region, a disconnect is issued. Accesses to undefined addresses will cause the PCI host to receive a Master Abort cycle (no DEVSEL# is asserted by the S5920).

For example, a PCI system may map a 512 byte Pass-Thru memory region to 0DC000h to 0DC1FFh. A PCI initiator attempts a four DWORD burst with a starting address of 0DC1F8h. The first and second data phases complete (filling the DWORDs at 0DC1F8h and 0DC1FCh), but the third data phase causes the S5920 to issue a disconnect. This forces the initiator to present the address 0DC200h on the PCI bus. If this address is part of another S5920 Pass-Thru region, the device accepts the access, but if not, a Master Abort cycle occurs.

PCI Write Disconnect

When the S5920 issues a disconnect for a PCI Pass-Thru write, it indicates that the Add-On is still completing a previous non-sequential Pass-Thru access or the FIFO is full. If the incoming access is a continuation of a previous one, no disconnect is issued and the transaction can continue where it left off (perhaps due to a previous disconnect or master time-out). PCI Operation Registers may be accessed while the Add-On is still completing a Pass-Thru access. Only Pass-Thru region accesses receive disconnect requests.

PCI Read Disconnect

If the S5920 issues a disconnect for a PCI Pass-Thru read, this indicates that the Add-On could not complete the read in the required time (16 clocks for the first data phase, 8 PCI clocks for the 2nd or later data phases).

When the PCI performs a read to a Pass-Thru region, the Add-On device must complete a Pass-Thru data transfer by writing the appropriate data into the Pass-Thru Data FIFO (APTD). If the Add-On can perform this before the required time (see above), the S5920 asserts TRDY# to complete a PCI read transfer. If the Add-On cannot complete the access within 16 clocks, a retry is requested (STOP# asserted without data transfer). If the Add-On manages to complete the data transfer into the PT Read FIFO, but after a retry was issued, the data is held in the FIFO until the original master comes back to read it. All subsequent PCI accesses to a Pass-Thru address other than the one corresponding to the data in the FIFO will be terminated with a PCI retry. Only a PCI access with a matching address can access the data in the PT Read FIFO, and thus release the Pass-Thru region for other accesses.

If the Add-On is busy performing a Pass-Thru write operation when a PCI read occurs, the S5920 requests an immediate retry. If the Add-On is busy performing a Pass-Thru read operation when another PCI read occurs, the S5920 determines whether the read is a retry from a previous access, and if so, attempts to continue the read where it left off. If the address is non-sequential, the new access is issued a retry. This allows the PCI bus to perform other operations. S5920 PCI Operation Registers may be accessed while the Add-On is still completing a Pass-Thru access. Only other Pass-Thru region accesses receive retry requests.

If the prefetch feature is enabled, the Pass-Thru interface will prefetch data, which should improve the performance on subsequent cycles to the same region. In the event that the Add-On cannot prefetch the first data before the S5920 issues a PCI retry, the prefetched data will be held in the read FIFO until the original master comes back to request it. Other PCI read requests to the Pass-Thru region will be terminated with immediate Retries.

If the prefetch feature is disabled, a PCI read cycle is not completed until the data is first transferred from the Add-On bus into the PT Read FIFO. The device will not prefetch, but will only request data from the Add-On bus after the PCI bus has requested the data. Pass-Thru bursts will not be performed in this case**.** In the event that a non-prefetchable Add-On cannot provide the second (or third, or fourth...) data to the PCI read request within the PCI Target Subsequent Latency period (eight PCI clocks), the S5920 will issue a PCI disconnect (STOP# asserted with data transfer). If the Add-On manages to transfer the second (or third, or fourth...) data to the PT Read FIFO, but after the disconnect, the data *may* be held in the FIFO until the original master comes back to read it. Depending upon the setting of the Retry Flush Enb bit, the data is held in the FIFO, and all other PCI read requests will be terminated with immediate Retries.

S5920 PASSIVE MODE OPERATION

The Pass-Thru address and data registers can be accessed as Add-On operation registers. The Pass-Thru FIFO is updated on the rising edge of ADCLK. For this reason, all Pass-Thru inputs must be synchronous to ADCLK. In the following sections the Add-On Pass-Thru interface is described for Pass-Thru single cycle accesses, burst accesses, target-requested retries, and when using 8-bit and 16-bit Add-On data buses.

Single-Cycle PCI to Pass-Thru Write

A single-cycle Pass-Thru write operation occurs when a PCI initiator writes a single value to the Pass-Thru region. PCI single cycle transfers consist of an address phase followed by one data phase. During the address phase of the PCI transfer, the S5920 stores the PCI address into the Pass-Thru Address Register (APTA). If the S5920 determines that the address is within one of its defined Pass-Thru regions, it captures the PCI data into the FIFO.

Figure 4 shows a single cycle Pass-Thru write access in the Passive Mode. The Add-On must read the data stored in the FIFO and transfer it to its destination. If the proper SELECT#, ADR[6:2] and BE[3:0]# signals are present, the S5920 will drive data one clock after RD# is asserted. It will stop driving data after the rising edge of ADCLK when RD# has been sampled deasserted.

Clock 0: The PCI bus cycle address information is stored in the S5920 and later stored in the Pass-Thru Address Register. The PCI address is recognized as a write to Pass-Thru region 1. The PCI data is stored in the S5920 Write FIFO.

Clock 1: Pass-Thru signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] are driven to indicate what action is required by Add-On logic. These status signals are valid only when PTATN# is active. Add-On logic can decode status signals upon the assertion of PTATN#.

PTATN# Asserted. Indicates a Pass-Thru access is pending

PTBURST# Deasserted. The access has a single data phase.

PTNUM[1:0] 1h. Indicates the access is to Pass-Thru region 1.

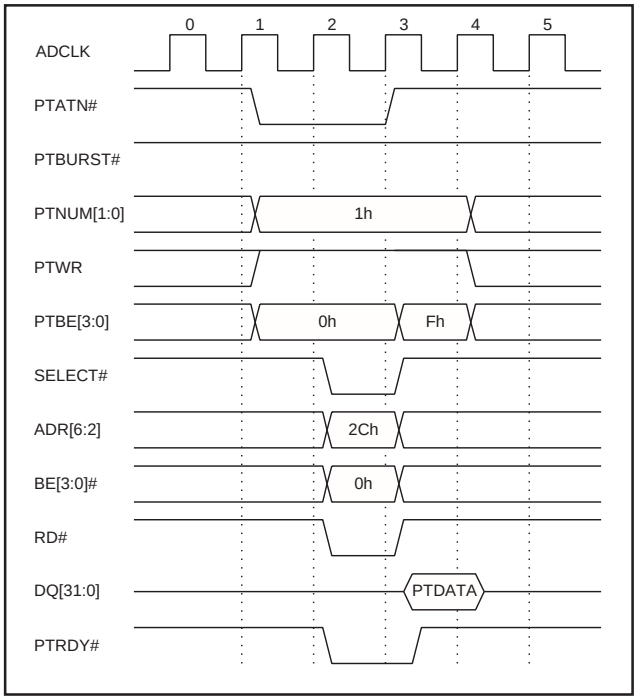
PTWR Asserted. The Pass-Thru access is a write.

PTBE[3:0]# 0h. Indicates the Pass-Thru access has all bytes valid.

Clock 2: SELECT#, ADR[6:2] and BE[3:0]# inputs are driven to read the Pass-Thru Write FIFO at offset 2Ch. DQ[31:0] is driven one clock after RD# and SELECT# are asserted. PTRDY# is asserted, indicating that the transfer is complete.

Clock 3: PTBE[3:0] will update one clock after RD# is asserted to indicate which bytes have not yet been read. The data is also driven on the DQ bus since RD# was asserted a clock earlier. Since PTRDY# was sampled asserted, PTATN# is deasserted and the Pass-Thru access is complete. If the Add-On logic requires more time to complete the read, PTRDY# can be delayed, extending the Pass-Thru cycle.

Figure 4. PCI To Add-On Passive Write

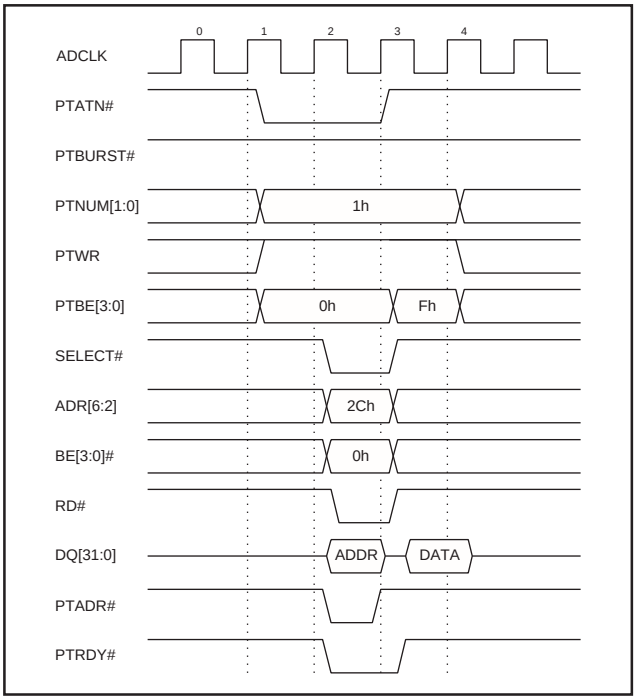


Clock 4: As PTATN# is deasserted, the Pass-Thru access is complete, and the S5920 can accept new Pass-Thru accesses starting on the next clock. The other Pass-Thru signals can also change state (in anticipation of a new transfer). The S5920 stops driving the DQ bus as RD# and SELECT# were not valid on the previous cycle.

Figure 5 shows a single cycle Pass-Thru write for the Passive Mode using the Pass-Thru address information. This provides PCI cycle address information to select a specific address location within an Add-On memory or peripheral. Add-On logic may latch the address for use during the data transfer (if PTADR# was asserted). Typically, the entire 32-bit address is not required. The Add-On may implement a scheme where only the required number of address bits are latched. It may also be useful to use the Pass-Thru region identifiers, PTNUM[1:0], as address lines. For example, Pass-Thru region 1 might be a 64K block of SRAM for data, while Pass-Thru region 2 might be 64K of SRAM for code storage (downloaded from the host during initialization). Using PTNUM0 as address line A16 allows two unique add-on memory regions to be defined.

Unlike all other Add-On operation register reads, the Add-On PTADR# input directly accesses the Pass-Thru Address Register and drives the contents onto the data bus during the same clock cycle. *WR# must not be asserted the same time as PTADR#, or there would be contention on the DQ bus!* However, it is permitted to assert RD# and PTADR# during the same cycle. This is because all reads performed with RD# are pipelined, while address reads with PTADR# are not pipelined.

Figure 5. PCI To Add-On Passive Write w/Pass-Thru Address



Clock 0: The address is recognized as a PCI write to Pass-Thru region 1. The PCI bus write address is stored in the Pass-Thru Address Register. The PCI bus write data is stored in the S5920 Write FIFO. Add-On bus signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] will update on the next rising edge of ADCLK.

Clock 1: Pass-Thru signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] are driven to indicate what action is required by Add-On logic. These status signals are valid only when PTATN# is active. Add-On logic can decode status signals upon the assertion of PTATN#.

PTATN#	Asserted. Indicates Pass-Thru access is pending.
PTBURST#	Not asserted. The access has a single data phase.
PTNUM[1:0]	1h. Indicates the access is to Pass-Thru region 1.
PTWR	Asserted. The Pass-Thru access is a write.
PTBE[3:0]#	0h. Indicate the Pass-Thru has all bytes valid.

Clock 2: The PTADR# input is asserted to read the Pass-Thru Address Register. The assertion of PTADR# will immediately cause the address to be driven on the DQ bus. RD#, SELECT#, byte enable, and the address inputs are asserted to read the Pass-Thru Data Register at offset 2Ch. DQ[31:0] is driven one clock after RD# and SELECT# are asserted. Asserting PTADR# and RD# at the same time will save a clock cycle, since the assertion of the RD# won't cause the data to be driven until a clock later. The Add-On also asserts PTRDY#, indicating that the current transfer is complete.

Clock 3: PTBE[3:0] are updated to indicate which bytes have not yet been read. Data is driven on the DQ bus because RD# was asserted a clock earlier. The Add-On logic reads the data and deasserts PTRDY#. As PTRDY# was sampled asserted, PTATN# is immediately deasserted and the Pass-Thru access is completed with the next clock. If add-on logic requires more time to read the Pass-Thru Data Register (slower memory or peripherals), PTRDY# can be delayed, extending the cycle.

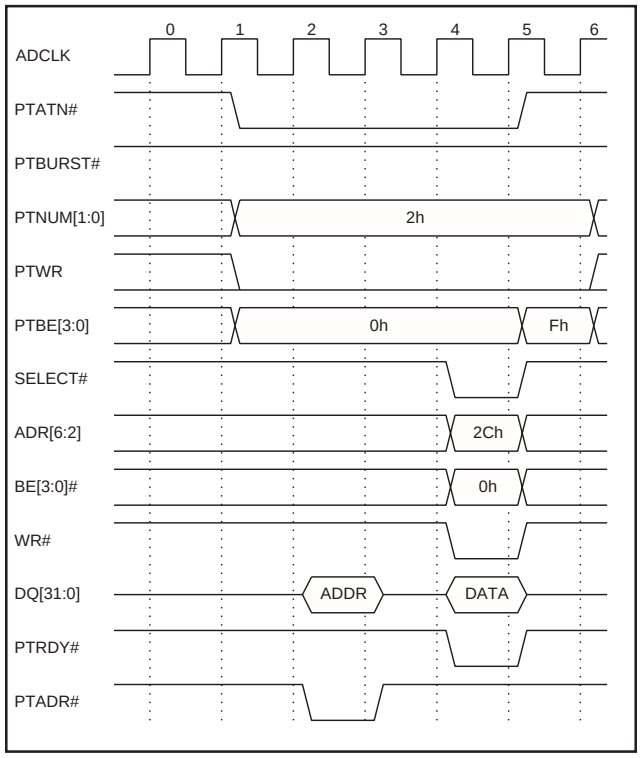
Clock 4: As PTATN# is deasserted, the Pass-Thru access is complete, and the S5920 can accept new Pass-Thru accesses starting on the next clock. The other Pass-Thru signals can also change (in anticipation of a new transfer). The S5920 stops driving the DQ bus as RD# and SELECT# were not valid on the previous cycle.

Single-Cycle PCI to Pass-Thru Read

A single-cycle PCI to Pass-Thru read operation occurs when a PCI initiator reads a single value from a Pass-Thru region. PCI single cycle transfers consists of an address phase followed by a single data phase. If the S5920 determines that the address is within one of its defined Pass-Thru regions, it indicates to the Add-On a write to the Pass-Thru Data Register (APTD) is required.

Figure 6 shows a Passive Mode single cycle Pass-Thru read access (Add-On write) using PTADR#. The Add-On reads data from a source on the Add-On and writes it to the APTD register.

Figure 6. PCI To Add-On Passive Read



Clock 0: The address is recognized as a PCI read of Pass-Thru region 2. The PCI bus read address is stored in the Pass-Thru Address Register. Add-On bus signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] will update on the next rising edge of ADCLK.

Clock 1: Pass-Thru signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] are driven to indicate what action is required by Add-On logic. These status signals are valid only when PTATN# is active. Add-On logic can decode status signals upon the assertion of PTATN#.

PTATN# Asserted. Indicates a Pass-Thru access is pending.

PTBURST# Deasserted. The access has a single data phase.

PTNUM[1:0] 2h. Indicates the access is to Pass-Thru region 2.

PTWR Deasserted. The Pass-Thru access is a read.

PTBE[3:0]# 0h. Indicates the Pass-Thru access has all bytes valid.

Clock 2: The PTADR# input is asserted to read the Pass-Thru Address Register. The address can be latched on the next rising-edge of ADCLK.

Clock 3: This turn-around cycle is required to avoid contention on the DQ bus. Time must be allowed after PTADR# is deasserted for the DQ outputs to float before add-on logic attempts to write to the Pass-Thru Read FIFO.

Clock 4: WR#, SELECT#, BE[3:0]#, and ADR[6:2] are asserted to write to the Pass-Thru Read FIFO at address 2Ch. The Add-On logic drives the DQ bus with the requested data. PTRDY# is also asserted, indicating that the Add-On is finished with the transfer.

Clock 5: The data on the DQ bus is latched into the Pass-Thru Read FIFO. As the S5920 samples PTRDY# asserted, PTATN# is deasserted and the Pass-Thru access is complete. PTBE[3:0] will update one clock after WR# is asserted to indicate which bytes have not yet been read. If add-on logic requires more time to provide data (slower memory or peripherals), PTRDY# can be delayed, extending the cycle.

Clock 6: As PTATN# is deasserted, the Pass-Thru access is complete, and the S5920 can accept new Pass-Thru accesses starting on the next clock. The other Pass-Thru signals can also change (in anticipation of a new transfer).

PCI to Pass-Thru Burst Writes

A PCI to Pass-Thru burst write operation occurs when a PCI initiator writes multiple values to a Pass-Thru region. A PCI burst cycle consists of an address phase followed by multiple data phases. If the S5920 determines that the requested address is within one of its defined Pass-Thru regions, the initial PCI address is stored into the Pass-Thru Address Register (APTA). The data from each data-cycle is individually latched into the Pass-Thru Data register (APTD) or Pass-Thru Write FIFO.

Figure 7. PCI to Add-On Passive Burst Write

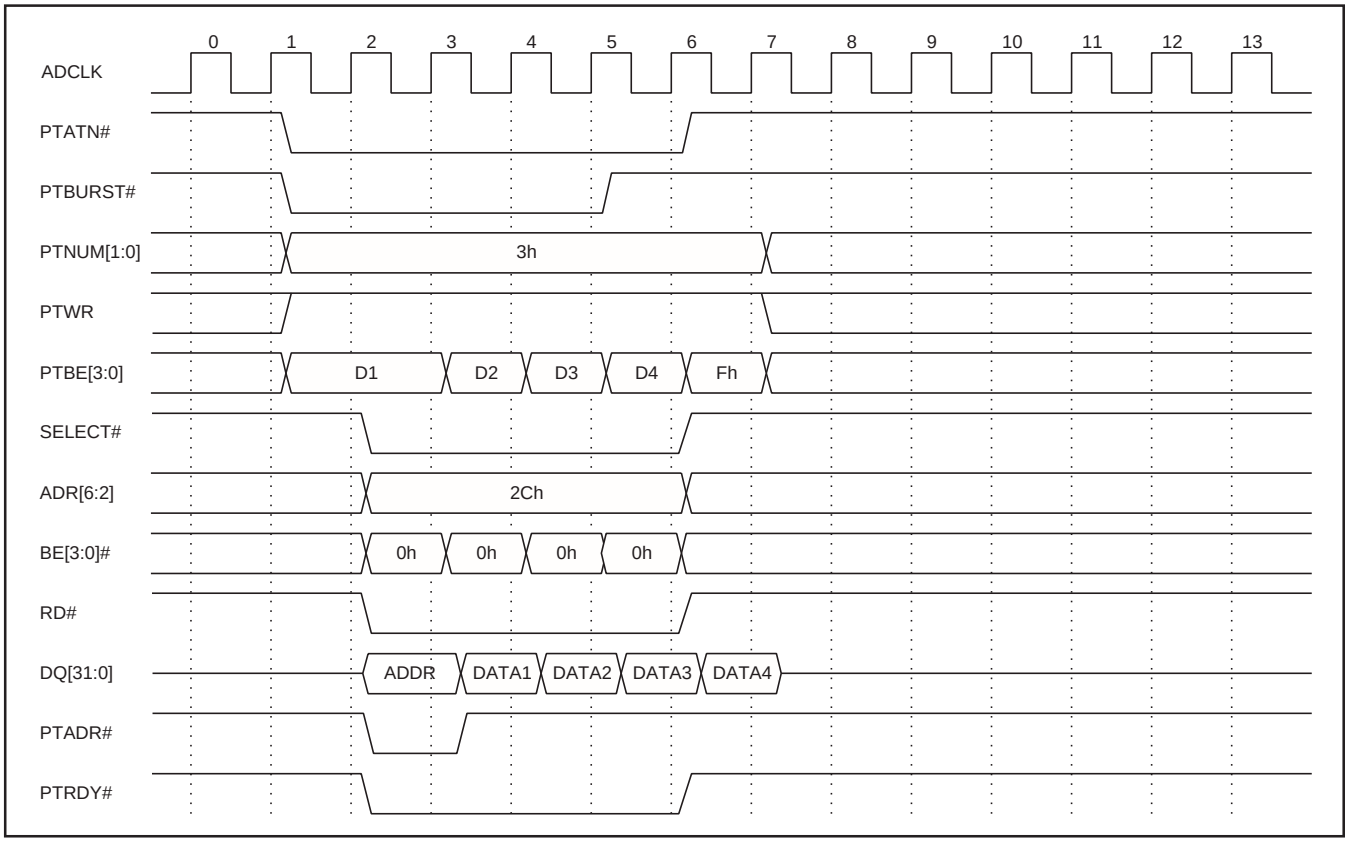


Figure 7 shows a Passive mode PCI to Add-On burst write of four DWORDs. In the following example, Add-On logic incorporates the use of PTADR# followed by multiple data reads to the S5920. If Add-On logic does not support burst accesses, PTADR# can be pulsed for individual data reads. The S5920 automatically increments the address in the APTA register during PCI bursts. In this example PTRDY# is continually asserted, indicating that Add-On logic is capable of accepting one DWORD per clock cycle. In addition, the PTBE[3:0] signals indicate a unique byte-enable for each data transfer.

The Pass-Thru Write FIFO (or APTD) can be disabled for bursts (do not accept PCI posted writes). In this case, the PCI is allowed to write to only one FIFO location and cannot continue bursting until the add-on has read the data. PTBURST# is never asserted when the PCI write FIFO is disabled. For this example, the Write FIFO is enabled.

Clock 0: The address is recognized as a PCI write to Pass-Thru region 1. The PCI bus write address is stored in the Pass-Thru Address Register. The PCI bus write data is stored in the S5920 write FIFO. Add-On bus signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] will update on the next rising edge of ADCLK.

Clock 1: Pass-Thru signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] are driven to indicate what action is required by Add-On logic. These status signals are valid only when PTATN# is active. Add-On logic can decode status signals upon the assertion of PTATN#.

PTATN#	Asserted. Indicates Pass-Thru access is pending.
PTBURST#	Asserted. The access has multiple data phases.
PTNUM[1:0]	3h. Indicates the access is to Pass-Thru region 3.
PTWR	Asserted. Indicates the access is a write.
PTBE[3:0]#	D1. Indicates valid bytes for the first data transfer.

Clock 2: The PTADR# input is asserted to read the Pass-Thru Address Register. The RD#, BE#, ADR[6:2] and SELECT# inputs are driven during this clock to read the Pass-Thru Data Register contents onto the DQ bus during the next clock. PTRDY# is asserted, indicating that the first transfer is complete.

Clock 3: The Add-On latches the address. Data 1 is driven on the DQ bus as a result of the previous read. As PTRDY# is sampled asserted, the PTBE# outputs are updated to indicate which bytes are valid for the second transfer. The BE[3:0]#, ADR[6:2], and SELECT# inputs remain driven along with RD# to read out the next data. PTRDY# remains asserted, indicating that the second transfer is complete.

Clock 4: Add-On logic uses the rising edge of this clock to store DATA1. DATA2 is driven on the DQ bus as a result of the previous read. As PTRDY# is sampled asserted, the PTBE# outputs are updated to indicate which bytes are valid for the third transfer. PTRDY# remains asserted, indicating that the third transfer is complete.

Clock 5: Add-On logic uses the rising edge of this clock to store DATA2. PTBURST# is deasserted indicating that only a single data phase remains. DATA3 is driven on the Add-On bus. The PTBE# outputs are updated to indicate which bytes are valid for the last transfer. PTRDY# remains asserted, indicating that the current transfer is complete.

Clock 6: Add-on logic uses the rising edge of this clock to store DATA3 from the S5920. PTRDY# sampled completes the last data phase. As a result, the S5920 deasserts PTATN#, and drives DATA4 onto the DQ bus. As the Add-on sampled PTBURST# deasserted and PTATN# asserted, it recognizes that the previous read was the last one. As a result, the Add-On deasserts SELECT#, ADR[6:2], BE[3:0]#, RD# and PTRDY#.

Clock 7: The Add-on logic stores DATA4 on the rising edge of this clock. As PTATN# is deasserted, the Pass-Thru access is complete, and the S5920 can accept new Pass-Thru accesses starting on the next clock. The other Pass-Thru signals can also change state (in anticipation of a new transfer).

Figure 8 illustrates a Passive mode transfer with a burst of five DWORDs in a PCI to Pass-Thru burst write with PTRDY# used to insert wait states. In some applications, Add-On logic may not be required to transfer data on every ADCLK and can use PTRDY# to control the data rate transfer. In this example, Add-On logic latches data every other clock cycle. RD# is shown deasserted when PTRDY# is deasserted, but could remain active during the entire Add-On burst. In this case, the DQ would not go to tri-state between reads, and the PTBE# outputs would lose some of their significance (as they would transition one cycle early as a result of the “unused” read).

Clock 0: The address is recognized as a PCI write to Pass-Thru region 0. The PCI bus write address is stored in the Pass-Thru Address Register. The PCI bus write data is stored in the S5920 write FIFO. Add-On bus signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] will update on the next rising edge of ADCLK.

Figure 8. PCI to Add-On Passive Burst Write Using PTRDY# to assert Wait-States

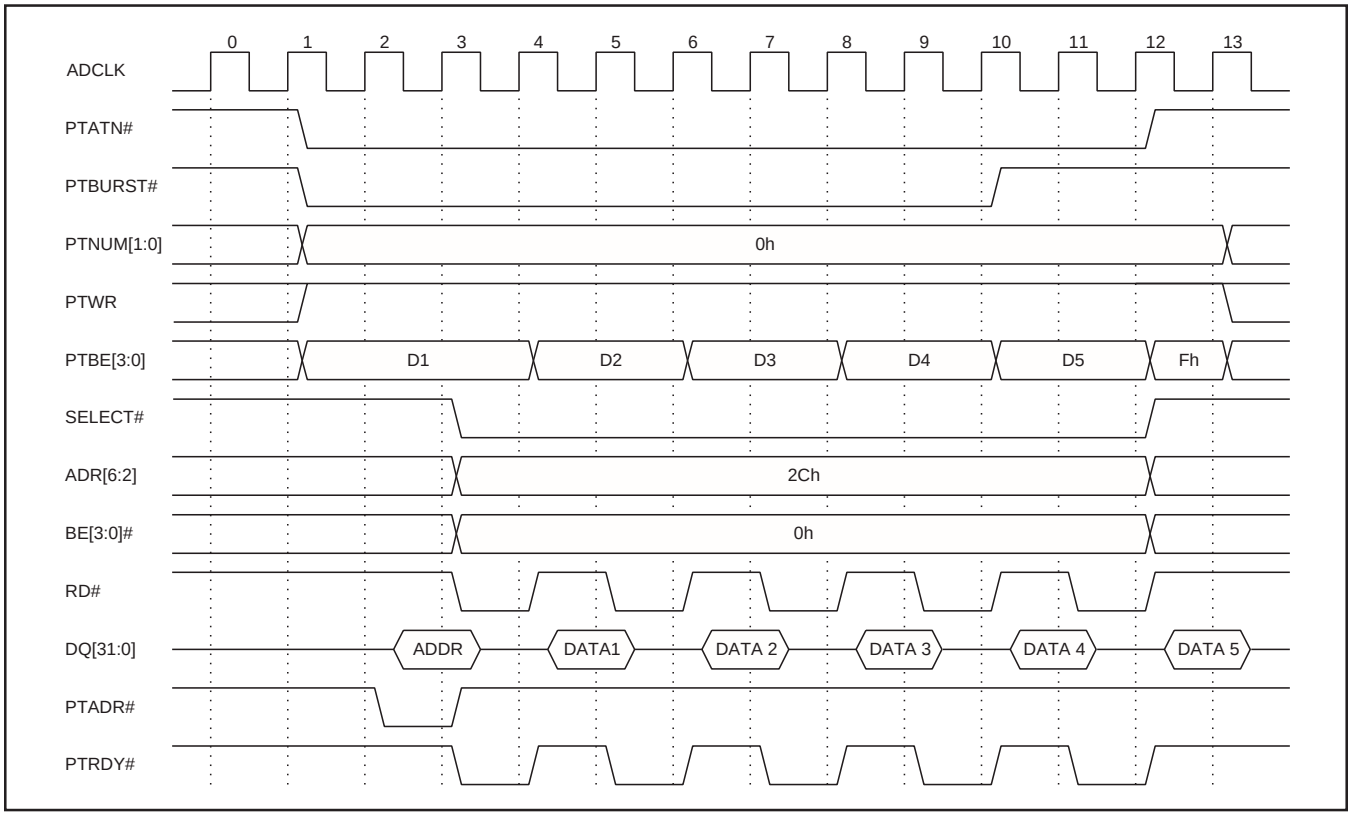
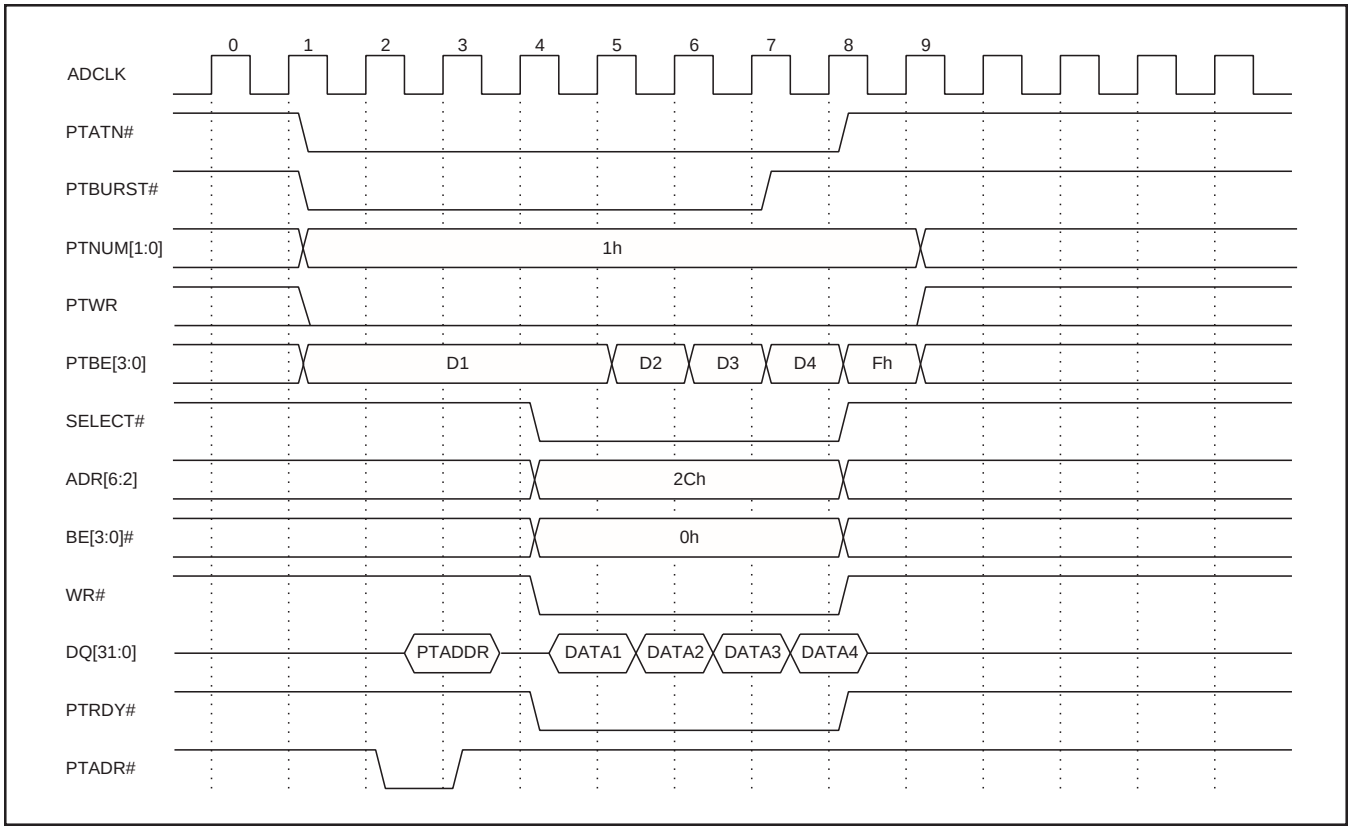


Figure 9. PCI to Add-On Passive Burst Read Access



Clock 1: Pass-Thru signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] are driven to indicate what action is required by Add-On logic. These status signals are valid only when PTATN# is active. Add-On logic can decode status signals upon the assertion of PTATN#.

PTATN#	Asserted. Indicates Pass-Thru access is pending.
PTBURST#	Asserted. The access has multiple data phases.
PTNUM[1:0]	0h. Indicates the access is to Pass-Thru region 3.
PTWR	Asserted. Indicates the access is a write.
PTBE[3:0]#	D1. Indicates valid bytes for the first data transfer.

Clock 2: Add-On logic samples PTATN# and PTBURST# asserted, indicating the start of a burst. The Add-On asserts PTADR# to read the Pass-Thru Address Register. As it is not ready to receive any data yet, it does not initiate a data read.

Clock 3: Add-on logic latches the address. RD#, BE[3:0]#, ADR[6:2], and SELECT# inputs are asserted to select the Pass-Thru Data Register during the next clock. PTRDY# is also asserted to indicate the completion of the first data phase.

Clock 4: As the S5920 sampled PTRDY# asserted, the first data phase is completed DATA1 is driven on the DQ bus, a result of the read from the previous clock cycle. The PTBE# outputs are updated to indicate which bytes are valid for the second transfer. Add-on logic is not fast enough to store the next data, so a wait state is activated by deasserting PTRDY#. RD# is also deasserted.

Clock 5: Add-On logic uses the rising edge of this clock to store DATA1. PTRDY# is sampled deasserted, so a wait state is activated. PTRDY# is asserted to indicate that the Add-On is ready to accept the next data transfer. RD# is also asserted, requesting DATA2 to be driven during the next clock cycle.

Clock 6: PTRDY# is sampled asserted, thus completing the current data-phase. DATA2 is driven on the DQ bus, a result of a read during the previous clock cycle. The PTBE# outputs are updated to indicate which bytes are valid for the third transfer. Add-on logic is not fast enough to store the next data, so a wait state is activated by deasserting PTRDY#. RD# is also deasserted.

Clock 7: Add-On logic uses the rising edge of this clock to store DATA2. PTRDY# is sampled deasserted, so a wait state is activated. PTRDY# is asserted to indicate that the Add-On is ready to accept the next data transfer. RD# is also asserted, requesting DATA3 to be driven during the next clock cycle.

Clock 8: PTRDY# is sampled asserted, thus completing the current data-phase. DATA3 is driven on the DQ bus, a result of a read during the previous cycle. The PTBE# outputs are updated to indicate which bytes are valid for the fourth transfer. Add-On logic is not fast enough to store the next data, so a wait state is activated by deasserting PTRDY#. RD# is also deasserted.

Clock 9: Add-On logic uses the rising edge of this clock to store DATA3. PTRDY# is sampled deasserted, so a wait state is activated. PTRDY# is asserted to indicate that the Add-On is ready to accept the next data transfer. RD# is also asserted, requesting DATA4 to be driven during the next clock cycle.

Clock 10: PTRDY# is sampled asserted, thus completing the current data-phase. PTBURST# is deasserted, indicating that only one DWORD is left for transfer. DATA4 is driven on the Add-On DQ bus, a result of a read during the previous clock cycle. The PTBE# outputs are updated to indicate which bytes are valid for the last transfer. Add-On logic is not fast enough to store the next data, so a wait state is activated by deasserting PTRDY#. RD# is also deasserted.

Clock 11: Add-On logic uses the rising edge of this clock to store DATA4. PTRDY# is sampled deasserted, so a wait state is activated. PTRDY# is asserted to indicate that the add-on is ready to accept the last data transfer. The add-on knows this is the last transfer as it has sampled PTBURST# deasserted and PTATN# asserted. RD# is also asserted, requesting DATA5 to be driven during the next clock cycle.

Clock 12: PTRDY# is sampled asserted, indicating that the last transfer was completed. As a result, PTATN# is deasserted. As the Add-On has also finished its transfer, it deasserts RD#, SELECT#, BE[3:0]#. The last data, DATA5, is driven on the Add-On DQ bus.

Clock 13: Add-On logic uses the rising edge of this clock to store DATA5. As PTATN# is deasserted, the Pass-Thru access is complete, and the S5920 can accept new Pass-Thru accesses starting on the next clock. The other Pass-Thru signals can also change state (in anticipation of a new transfer).

Pass-Thru Burst Reads

A Pass-Thru burst read operation occurs when a PCI initiator reads multiple DWORDs from a Pass-Thru region. A burst transfer consists of a single address and multiple data phases. The S5920 stores the PCI address into the Pass-Thru Address Register (APTA). If the S5920 determines that the address is within one of its defined Pass-Thru regions, it indicates to the Add-On that a write to the Pass-Thru Data Register or Pass-Thru Read FIFO (APTD) is required.

Figure 10. PCI to Add-On Passive Burst Read

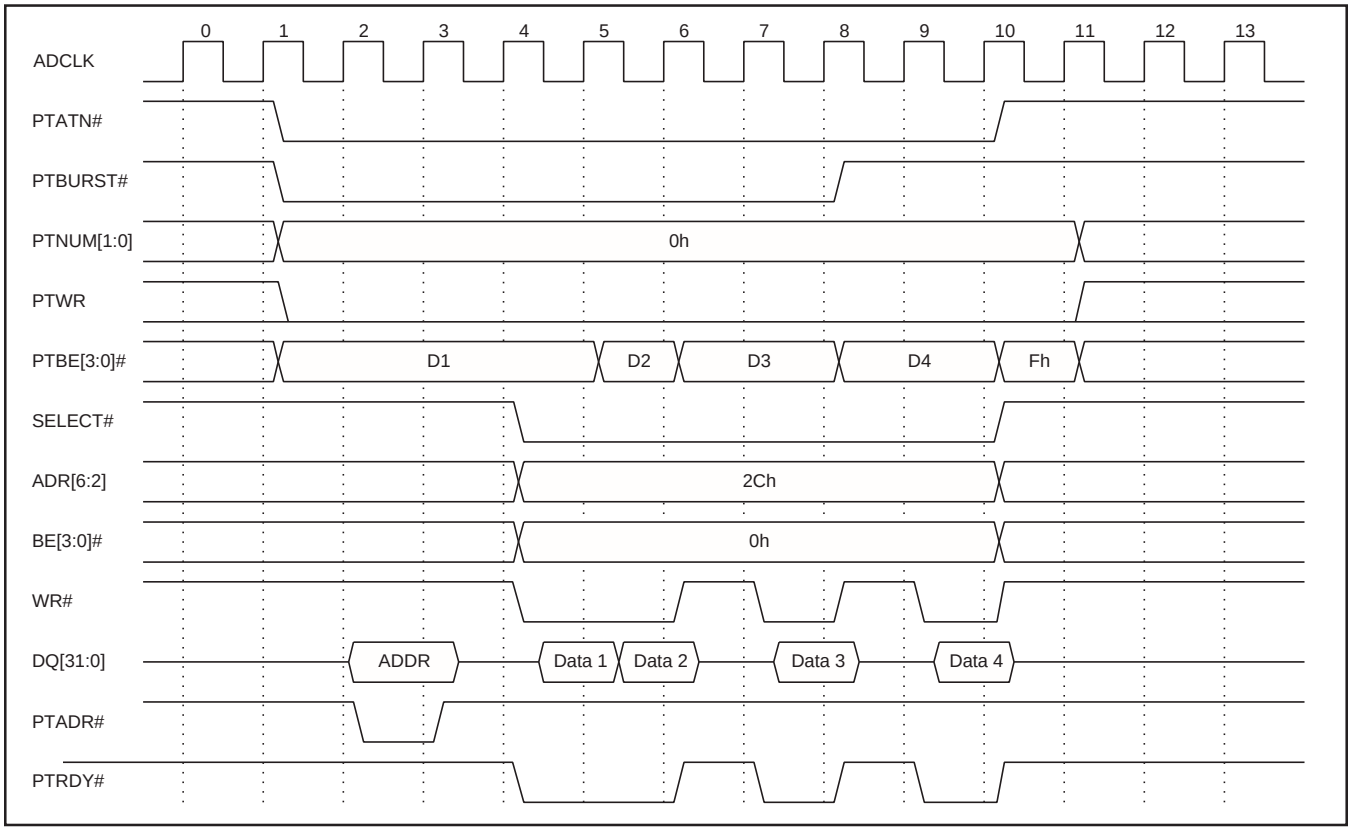


Figure 9 shows a Passive Mode Pass-Thru burst read access (Add-On write) of four DWORDs, using PTADR# to provide an address-phase.

Clock 0: PCI address information is stored in the Pass-Thru Address Register. The address is recognized as a PCI read of Pass-Thru region 1. Add-On bus signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] will update on the next rising edge of ADCLK

Clock 1: Pass-Thru signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] are driven to indicate what action is required by Add-On logic. These status signals are valid only when PTATN# is active. Add-On logic can decode status signals upon the assertion of PTATN#.

PTATN#	Asserted. Indicates Pass-Thru access is pending.
PTBURST#	Asserted. The access has multiple data phases.
PTNUM[1:0]	1h. Indicates the access is to Pass-Thru region 1.
PTWR	Deasserted. Indicates the access is a read.
PTBE[3:0]#	D1. Indicates valid bytes for the first data transfer.

Clock 2: The Add-On logic has sampled PTATN# and PTBURST# active, indicating that at least two read data transfers are requested by the PCI. The Add-On will start servicing the Burst Read transfer by first reading the Pass-Thru Address via the PTADR# input. This is an asynchronous read, meaning that the address will appear on DQ after a propagation delay from the assertion of PTADR#. In the event that the address is not required, this cycle and the next could be skipped (as the next clock provides a turn-around cycle).

Clock 3: The Add-On logic will latch the Pass-Thru address on the rising edge of this clock. This cycle is also required to avoid contention on the DQ bus. Time must be allowed after PTADR# is deasserted for the DQ outputs to float before Add-On logic attempts to write to the Pass-Thru Read FIFO.

Clock 4: The BE[3:0]#, ADR[6:2], and SELECT# inputs are asserted. WR# and DQ are asserted, indicating that DATA1 is to be written to the PT Read FIFO on the next clock. PTRDY# is asserted, to indicate the completion of the current data phase.

Clock 5: As the S5920 samples WR# asserted, it writes DATA1 into the PT Read FIFO. PTRDY# is sampled asserted, which completes the first data transfer and updates the internal FIFO pointers. The PTBE# outputs are updated to indicate which bytes are valid for the second transfer. The Add-On logic samples

PTBURST# asserted, so it knows more data is being requested. The Add-On keeps WR# asserted, and drives DATA2 onto the DQ bus. PTRDY# is also asserted to complete the current data phase.

Clock 6: As the S5920 samples WR# asserted, it writes DATA2 into the PT Read FIFO. PTRDY# is sampled asserted, which completes the second data transfer and updates the internal FIFO pointers. The PTBE# outputs are updated to indicate which bytes are valid for the third transfer. The Add-On logic samples PTBURST# asserted, so it knows more data is being requested. The Add-On keeps WR# asserted, and drives DATA3 onto the DQ bus. PTRDY# is also asserted to complete the current data phase.

Clock 7: As the S5920 samples WR# asserted, it writes DATA3 into the PT Read FIFO. PTRDY# is sampled asserted, which completes the third data transfer and updates the internal FIFO pointers. The PTBE# outputs are updated to indicate which bytes are valid for the last transfer. The S5920 deasserts PTBURST#, indicating that the previous read was the second to last. The next transfer from the Add-On bus will be the last. The Add-On logic samples PTBURST# asserted, so it knows more data is being requested. The Add-On keeps WR# asserted, and drives DATA4 onto the DQ bus. PTRDY# is also asserted to complete the current data phase.

Clock 8: As the S5920 samples WR# asserted, it writes DATA4 into the PT Read FIFO. PTRDY# is sampled asserted, which completes the final data transfer and updates the internal FIFO pointers. The S5920 deasserts PTATN#, indicating that the final transfer was performed. No more data is being requested from PCI. The Add-On logic samples PTBURST# deasserted, so it knows that the previous data transfer was the last, and no more data is being requested. The Add-On deasserts WR#, ADR[6:2], SELECT#, BE[3:0]# and DQ. It also deasserts PTRDY#. Note that in a synchronous design, the Add-On logic does not require PTATN# in order to terminate a Pass-Thru read operation, PTBURST# is used for this.

Clock 9: As PTATN# and PTBURST# are deasserted, the Pass-Thru access is complete, and the S5920 can accept new Pass-Thru accesses starting on the next clock. The other Pass-Thru signals can also change state (in anticipation of a new transfer).

NOTE: With prefetch disabled, the performance of Pass-Thru burst reads will be less than optimal. Because of certain issues involving synchronizing signals across clock boundaries (ADCLK -> PCLK), Pass-Thru burst reads will occur only in double and single accesses. For example, a Pass-Thru burst read of five data phases would translate to a burst-read of two DWORDs, another burst-read of two DWORDs followed by a single burst-read with PTATN# being deasserted between each burst packet, losing potentially valuable clock cycles. It is recommended to enable prefetch if maximum performance is desired.

Figure 10 also shows a Passive Mode Pass-Thru burst read, but the Add-On logic uses PTRDY# to control the rate at which data is transferred. In many applications, Add-On logic is not fast enough to provide data every ADCLK. In this example, the Add-On interface writes data every other clock cycle.

Using PTRDY# to assert Wait-States

Clock 0: PCI address information is stored in the Pass-Thru Address Register. The address is recognized as a PCI read of Pass-Thru region 1. Add-On bus signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] will update on the next rising edge of ADCLK.

Clock 1: Pass-Thru signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] are driven to indicate what action is required by Add-On logic. These status signals are valid only when PTATN# is active. Add-On logic can decode status signals upon the assertion of PTATN#.

PTATN#	Asserted. Indicates Pass-Thru access is pending..
PTBURST#	Asserted. The access has multiple data phases.
PTNUM[1:0]	0h.Indicates. the access is to Pass-Thru region 0.
PTWR	Deasserted. Indicates the access is a read.
PTBE[3:0]#	D1. Indicates valid bytes for the first data transfer.

Clock 2: The Add-On logic has sampled PTATN# and PTBURST# active, indicating that at least two read data transfers are requested by the PCI. The Add-On will start servicing the Burst Read transfer by first reading the Pass-Thru Address via PTADR#. This is an asynchronous read, meaning that the address will appear on DQ after a propagation delay from the assertion of PTADR#. In the event that the address is not required, this cycle and the next could be skipped (as the next clock provides a turn-around cycle).

Clock 3: The Add-On logic will latch the Pass-Thru address on the rising edge of this clock. This cycle is also required to avoid contention on the DQ bus. Time must be allowed after PTADR# is deasserted for the DQ outputs to float before Add-On logic attempts to write to the Pass-Thru Read FIFO.

Clock 4: The BE[3:0]#, ADR[6:2], and SELECT# inputs are asserted. WR# and DQ are asserted, indicating that DATA1 is to be written to the PT Read FIFO on the next clock. PTRDY# is asserted, to indicate the completion of the current data phase.

Clock 5: As the S5920 samples WR# asserted, it writes DATA1 into the PT Read FIFO. PTRDY# is sampled asserted, which completes the first data transfer and updates the internal FIFO pointers. The PTBE# outputs are updated to indicate which bytes are valid for the second transfer. The Add-On logic samples PTBURST# asserted, so it knows more data is being requested WR# remains asserted, and DATA2 is driven onto DQ. PTRDY# is also asserted to complete the current data phase.

Clock 6: As the S5920 samples WR# asserted, it writes DATA2 into the PT Read FIFO. PTRDY# is sampled asserted, which completes the second data transfer and updates the internal FIFO pointers. The PTBE# outputs are updated to indicate which bytes are valid for the third transfer. The Add-On logic samples PTBURST# asserted, so it knows more data is being requested. However, it is not ready to transfer data yet, so it deasserts PTRDY# and WR#, and stop driving the DQ bus. The DQ bus could be in tri-state.

Clock 7: As the S5920 samples WR# and PTRDY# deasserted, no data was written to the PT Read FIFO and the FIFO pointer was not updated (as the transfer was not signaled complete via a PTRDY#). The Add-On logic is ready to continue the transfer, so it asserts WR# and drives the DQ bus with DATA3. PTRDY# is also asserted to complete the current data phase.

Clock 8: As the S5920 samples WR# asserted, it writes DATA3 into the PT Read FIFO. PTRDY# is sampled asserted, which completes the third data transfer and updates the internal FIFO pointers. The S5920 deasserts PTBURST#, indicating that the previous read was the second to last. The next transfer from the Add-On will be the last. The PTBE# outputs are updated to indicate which bytes are valid for the last transfer. The Add-On logic samples PTBURST# asserted, so it knows more data is being requested. However, it is not ready to transfer data yet, so it deasserts PTRDY# and WR#. The data on the DQ bus is a don't care, as the Add-On is not writing during this cycle. The DQ bus could be in tri-state.

Clock 9: As the S5920 samples WR# and PTRDY# deasserted, no data was written to the PT Read FIFO and the FIFO pointer was not updated (as the transfer was not signaled complete via a PTRDY#). The Add-On logic samples PTBURST# deasserted and PTATN# asserted, so it knows that the previous data transfer was the last, and no more data is being requested. However, as it inserted a wait state during the previous cycle, it still has one more transfer to complete. As the Add-On logic is ready to complete the transfer, it asserts WR# and drives the DQ bus with DATA4. PTRDY# is also asserted to complete the last data phase.

Clock 10: As the S5920 samples WR# asserted, it writes DATA4 into the PT Read FIFO. PTRDY# is sampled asserted, which completes the last data transfer and updates the internal FIFO pointers. The S5920 deasserts PTATN#, indicating that the final transfer was performed. No more data is being requested from PCI. Since the Add-On logic previously sampled PTBURST# deasserted, and transferred the last data, it knows that no more data is being requested. The Add-On deasserts WR#, ADR[6:2], SELECT#, BE[3:0]# and DQ. It also deasserts PTRDY#.

Clock 11: As PTATN# and PTBURST# are deasserted, the Pass-Thru access is complete, and the S5920 can accept new Pass-Thru accesses starting on the next clock. The other Pass-Thru signals can also change state (in anticipation of a new transfer).

8-Bit and 16-Bit Pass-Thru Add-On Bus Interface in Passive Mode

The S5920 allows a simple interface to devices with 8-bit or 16-bit data buses. Each Pass-Thru region may be defined as 8, 16 or 32 bits, depending on the contents of the boot device which is loaded into the PCI Base Address Configuration Registers during initialization. The result of the initialization is a unique bussize (8/16/32 bits) for each Pass-Thru region. The Pass-Thru Add-On interface internally controls byte lane steering to allow access to the 32-bit Pass-Thru Data FIFO (APTD) from 8-bit or 16-bit Add-On buses. The four DQ data bytes are internally steered depending upon the bus size of the region and the values of the Byte Enables (BE#). Note that this 8-/16-bit internal byte-lane steering is *not* performed for other Add-On operation registers, just the APTD register (ADR = 2Ch).

Table 1. Byte Lane Steering for PCI Write (Add-On Read)

Byte Enables				APTD Register Write Byte Lane Steering			
3	2	1	0	DQ[31:24]	DQ[23:16]	DQ[15:8]	DQ[7:0]
x	x	x	0	BYTE3	BYTE2	BYTE1	BYTE0
x	x	0	1	BYTE3	BYTE2	BYTE1	BYTE1
x	0	1	1	BYTE3	BYTE2	BYTE2	BYTE2
0	1	1	1	BYTE3	BYTE3	BYTE3	BYTE3

Table 2. Byte Lane Steering for PCI Read (Add-On Write)

Defined PT Bus Width	APTD Register Write Byte Lane Steering			
	BYTE3	BYTE2	BYTE1	BYTE0
32 Bit Data Bus	DQ[31:24]	DQ[23:16]	DQ[15:8]	DQ[7:0]
16 Bit Data Bus	DQ[15:8]	DQ[7:0]	DQ[15:8]	DQ[7:0]
8 Bit Data Bus	DQ[7:0]	DQ[7:0]	DQ[7:0]	DQ[7:0]

For Pass-Thru writes (Add-On APTD reads), Add-On logic must read the APTD register one byte or one word at a time (depending on the Add-On bus width). The internal data bus is steered from the correct portion of APTD using the BE[3:0]# inputs. Table 1 shows the byte lane steering mechanism used by the S5920. The BYTEn symbols indicate data bytes in the Pass-Thru Data Register.

When a read by the Add-On is performed with a BEn# input asserted, the corresponding PTBEn# output is deasserted. Add-On logic cycles through the byte enables to read the entire APTD Register. Once all data is read (all PTBE[3:0]#s are deasserted), PTRDY# is asserted by the Add-On, completing the access.

For Pass-Thru reads (Add-On APTD writes), the bytes requested by the PCI initiator are indicated by the PTBE[3:0]# outputs. Add-On logic uses the PTBE[3:0]# signals to determine which bytes must be written (and which bytes have already been written). For example, a PCI initiator performs a byte Pass-Thru read from an 8-bit Pass-Thru region with PCI BE2# asserted. On the Add-On interface, PTBE2# is asserted, indicating that the PCI initiator requires data on this byte lane. Once the Add-On writes APTD, byte 2, PTBE2# is deasserted, and the Add-On may assert PTRDY#, completing the cycle.

Table 2 shows how the external Add-On data bus is steered to the Pass-Thru Data Register bytes. This mechanism is determined by the Pass-Thru region bus width defined during initialization. The BYTEn symbols indicate data bytes in the Pass-Thru Data Register. For example, an 8-bit Add-On write with BE1# asserted results in the data on DQ[7:0] being steered into BYTE1 of the APTD register.

To write data into the APTD Register, PTBEn# and BEn# must both be asserted. The following describes how APTD writes are controlled:

Write BYTE3 if PTBE3# AND BE3# are asserted

Write BYTE2 if PTBE2# AND BE2# are asserted

Write BYTE1 if PTBE1# AND BE1# are asserted

Write BYTE0 if PTBE0# AND BE0# are asserted

After each byte is written into the Pass-Thru data register, its corresponding PTBE[3:0]# output is deasserted. This allows Add-On logic to monitor which bytes have been written, and which bytes remain to be written. When all requested bytes have been written (all PTBE[3:0]#s are deasserted), PTRDY# is asserted by the Add-On, completing the access.

There are two methods of accessing the Add-On Pass-Thru Address Register (APTA): by asserting the PTADR# pin (and getting the address on DQ after some propagation delay) or by asserting RD#, SELECT, BE[3:0]#'s, and ADR[6:2] = 28h (and getting the address on DQ one cycle later). When using the PTADR# input, all 32 bits of address are driven on DQ, regardless of the state of the DQMODE pin. When accessing APTA via an Add-On operation register access, all 32 bits of address are driven on DQ as long as DQMODE indicates 32 bits. If DQMODE is set for 16 bits, it is necessary to perform two accesses: one with BE[3]# low for the lower 16 bits, and one with BE[3]# high for the upper 16 bits. The Pass-Thru region bus-sizes have *no* effect on APTA accesses.

Figure 11 shows a Pass-Thru write operation for a region defined for an 8-bit Add-On bus interface. As the 8-bit device is connected only to DQ[7:0], the device must access the APTD one byte at a time.

A PCI initiator has performed a posted burst-write of two DWORDs to Pass-Thru region zero. Data0 = 08D49A30h and Data1 = AABBCCDDh. All byte-enables of the DWORDs were active.

Clock 0: The address is recognized as a PCI write to Pass-Thru region 0. The PCI bus write address is stored in the Pass-Thru Address Register. The PCI bus write data is stored in the S5920 write FIFO. Add-On bus signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] will update on the next ADCLK.

Clock 1: Pass-Thru signals PTATN#, PTBURST#, PTNUM[1:0], PTWR and PTBE[3:0] are driven to indicate what action is required by Add-On logic. These status signals are valid only when PTATN# is active. Add-On logic can decode status signals upon the assertion of PTATN#.

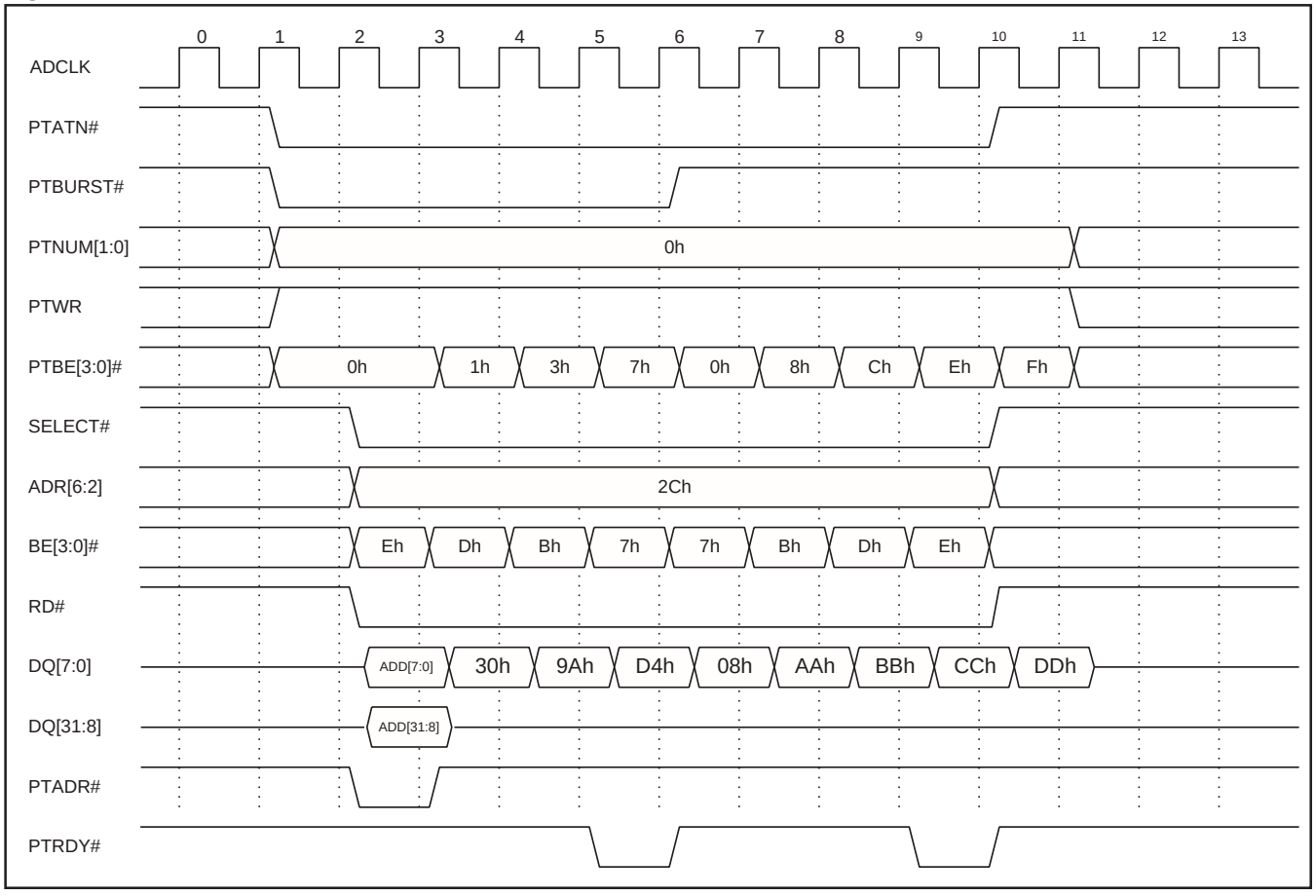
PTATN#	Asserted. Indicates Pass-Thru access is pending.
PTBURST#	Asserted. The access has multiple data phases.
PTNUM[1:0]	0h. Indicates the access is to Pass-Thru region 0.
PTWR	Asserted. Indicates the access is a write.
PTBE[3:0]#	0h. Indicates valid bytes for the first data transfer.

Clock 2: The Add-On sees that a burst-write is being requested by the PCI, so starts by reading the corresponding address via PTADR#. Note that all 32 bits of the APTA are output on the DQ bus when PTADR# is asserted. The Add-On must be capable of latching the upper 24 bits (if needed). The Add-On begins reading the APTD Register (asserting SELECT#, ADR[6:2], and RD#). The Add-On logic sees that all bytes are valid (PTBE# = 0h), so starts the read by asserting BE0#, to indicate that BYTE0 of the APTD is to be driven on DQ[7:0] during the next clock cycle.

Clock 3: The Add-On logic latches the Pass-Thru address. RD# and BE0# are sampled by the S5920, so BYTE0 of the APTD is driven on DQ[7:0] and PTBE0# is deasserted. The Add-On asserts RD# and BE1#, thus requesting that BYTE1 of the APTD be driven on the DQ bus during the next cycle.

Clock 4: The Add-On logic latches BYTE0. RD# and BE1# are sampled asserted by the S5920, so BYTE1 of the APTD is driven on DQ[7:0] and PTBE1# is deasserted. The Add-On device asserts RD# and BE2#, thus requesting that BYTE2 of the APTD be driven on the DQ bus during the next cycle.

Figure 11. PCI to Add-On Passive Write to an 8-bit Add-On Device



Clock 5: The Add-On logic latches BYTE1. RD# and BE2# are sampled asserted by the S5920, so BYTE2 of the APTD is driven on DQ[7:0] and PTBE2# is deasserted. The Add-On device asserts RD# and BE3#, thus requesting that BYTE3 of the APTD be driven on the DQ bus during the next cycle. PTRDY# is also asserted, indicating that the transfer is complete.

Clock 6: The Add-On logic latches BYTE2. RD# and BE3# are sampled asserted by the S5920, so BYTE3 of the APTD is driven on DQ[7:0]. PTRDY# is sampled asserted, so the previous transfer is complete. The PTBE# signals are updated to indicate which bytes are valid for the next transfer (in this case, all bytes are valid for the second DWORD, so PTBE# = 0h). The S5920 deasserts PTBURST#, as it only has one DWORD left to transfer. The Add-On device asserts RD# and BE3#, thus requesting that BYTE3 of the second DWORD in the APTD be driven on the DQ bus during the next cycle.

Clock 7: The Add-On logic latches BYTE3 of the first DWORD. RD# and BE3# are sampled asserted by the S5920, so BYTE3 of the second DWORD in the APTD is driven on DQ[7:0] and PTBE3# is deasserted. The Add-On device asserts RD# and BE2#, thus requesting that BYTE2 of the APTD be driven on the DQ bus during the next cycle.

Clock 8: The Add-On logic latches BYTE3 of the second DWORD. RD# and BE2# are sampled asserted by the S5920, so BYTE2 of the APTD is driven on DQ[7:0] and PTBE2# is deasserted. The Add-On asserts RD# and BE1#, thus requesting that BYTE1 of the APTD be driven on the DQ bus during the next cycle.

Clock 9: The Add-On logic latches BYTE2 of the second DWORD. RD# and BE1# are sampled by the S5920, so BYTE1 of the APTD is driven on DQ[7:0] and PTBE1# is deasserted. The Add-On asserts RD# and BE0#, thus requesting that BYTE0 of the APTD be driven on the DQ bus during the next cycle. PTRDY# is also asserted, indicating that the transfer is complete. As PTBURST# is already deasserted, the Add-On recognizes that this is the last transfer.

Clock 10: The Add-On logic latches BYTE1 of the second DWORD. RD# and BE0# are sampled by the S5920, so BYTE0 of the APTD is driven on DQ[7:0]. PTRDY# is sampled asserted, so the previous transfer is complete. The PTBE# signals are updated to indicate which bytes are valid for the next transfer (in this case, there is no more valid data to transfer, so PTBE = Fh). The S5920 deasserts PTATN#, as it has no data left to transfer. The Add-On device deasserts RD#, BE#, ADR[6:2], SELECT# as the data transfer is complete.

Clock 11: The Add-On logic latches BYTE0 of the second DWORD. PTATN# and PTBURST# both deasserted indicate that the Pass-Thru transfer is complete. The PCI can start another access on the next clock cycle.

For 16-bit peripheral devices, the byte steering works in the same way. Because the Add-On data bus is 16 bits wide, only two 16-bit cycles are required to access the entire APTD Register. Two byte enables can be asserted during each access.

Figure 12 shows a Pass-Thru read operation for a region defined for a 16-bit Add-On bus interface. As the 16-bit device is connected only to DQ[15:0], the device must access the APTD one word at a time. The Add-On must be capable of latching the upper 16 bits of the APTA (if they are needed).

The PCI initiator has requested a 32-bit burst read from Pass-Thru region three. All PTBE#s are asserted.

Clock 1: The Add-On begins by reading the APTA register (asserting PTADR#). All 32 bits of the address are driven on the DQ bus.

Clock 2: Turn-around cycle, preventing potential bus contention on the DQ bus.

Clock 3: The Add-On initiates the write by asserting WR#, SELECT#, BE[3:0]# = “1100”, ADR[6:2] = 2Ch and the low word of the first DWORD to be transferred (D0-LO).

Clock 4: The S5920 updates the PTBE#s to indicate that the low word was provided, and that the upper word is still required. The Add-On drives the upper word (D0-HI), and activates the appropriate byte enables, BE# = 0011 The Add-On also asserts PTRDY#, indicating that it is done with the current DWORD, and to advance the FIFO pointer and prepare for the second DWORD.

Clock 5: The PTBE#s are updated to indicate that the next DWORD to be transferred requires all bytes. The Add-On drives DQ[15:0] with the lower word of the second DWORD (D1-LO), and the byte-enables indicate the same, BE# = 1100. The Add-On also deasserts PTRDY#. This process continues until the transfer is complete and all words have been written.

Endian Conversion

Endian conversion can be enabled/disabled for each Pass-Thru Region. It is controlled by bits 6, 14, 22 and 30 of the PTCR. The default endian type for the S5920 is Little Endian. For this reason, the default values in the PTCR are for Little Endian. If Big Endian is selected, the Pass-Thru data and byte-enable interface will be converted to Big Endian type.

When the device is programmed for Big Endian translation and a 32-bit data bus, the S5920 will convert as described in Table 3.

S5920 ACTIVE MODE OPERATION

Active mode is provided to simplify logic requirements when interfacing an application to the Add-On Local bus. Passive mode requires Add-On logic to assert read/write signals and drive or latch data on the DQ bus.

Strapping PTMODE low configures the S5920 for Active mode operation. Active mode allows more designer flexibility through programmable features. The following is a brief description of these features.

- Pass-Thru address can be driven automatically at the beginning of all transfers or can be skipped altogether if addresses are unneeded by Add-On logic.
- Programmed or Add-On controlled wait states to delay data transfers automatically or on the fly.
- Endian Conversion
- Write FIFO (Write posting)
- Read FIFO (Prefetch)

Active Operation

In Active mode, a data transfer start is signaled on the first clock edge in which PTATN# is sampled low. If PTADR# has been programmed to be output it will go active (low) at this time, and the data presented on the DQ bus is the address for the current transaction. Add-On logic may latch the address value at the rising edge of the clock. Address cycles do not count toward the number of wait states needed to complete data phases. In Active mode, the PTRDY# pin is renamed to PTWAIT#. On cycles after PTWAIT# is sampled low, the state machine is idle. Idle cycles are also not

Table 3. Showing Big Endian Conversion for 32-bit

Byte#	PCI Byte	Add-On Byte
0	D7-D0	D31-D24
1	D15-D8	D23-D16
2	D23-D15	D15-D8
3	D31-D24	D7-D0

Figure 12. PCI to Add-On Passive Read to an 16-bit Add-On Device

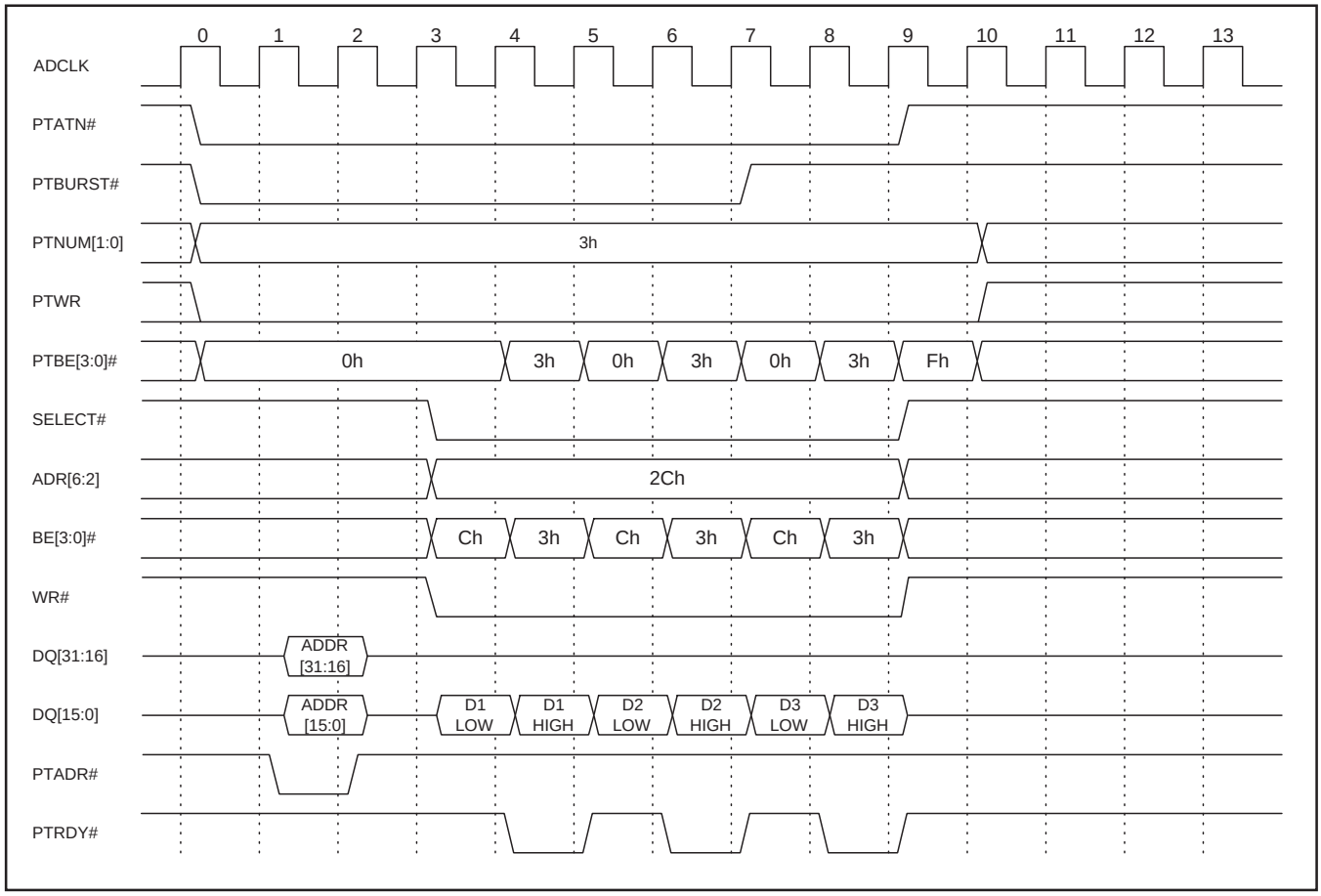


Table 4. Big Endian conversion for a 16-bit bus. The S5920 drives D[15:0] only

Transfer	Byte #	PCI Byte Lane	Add-On Bus Byte Lane
1st XFER	0	D7-D0	D15-D8
1st XFER	1	D15-D8	D7-D0
2nd XFER	2	D23-D16	D15-D8
2nd XFER	3	D31-D24	D7-D0

Table 5. Big Endian conversion for an 8-bit bus. The S5920 drives D[7:0] only

Transfer	Byte #	PCI Byte Lane	Add-On Bus Byte Lane
1st XFER	0	D7-D0	D7-D0
2nd XFER	1	D15-D8	D7-D0
3rd XFER	2	D23-D16	D7-D0
4th XFER	3	D31-D24	D7-D0

counted as wait states by the S5920. To control the number of wait states on an as-needed basis only, zero wait states should be programmed and PTWAIT# can be driven low when wait states are to be inserted. If PTWAIT# is low when PTATN# is asserted by the S5920, the pending transfer cycle won't be started until PTWAIT# is driven high.

In Active mode, wait states can also be programmed. This enables easier interfacing to slow Add-On logic which cannot transfer data at the full ADCLK speed. The S5920 inserts a turnaround cycle after the address phase for PCI Read cycles. If one or more wait states have been programmed, the turnaround cycle is considered the first wait state of the first data phase of that transaction.

For all Active mode transfers, the DXFR# signal is used by Add-On logic as the data transfer signal. Data must be latched at the rising edge of ADCLK when DXFR# is asserted for a PCI write. Conversely, for PCI Reads, the rising edge of ADCLK when DXFR# is asserted can be used to increment to the next data field.

Figure 13a. Active mode PCI Read (Zero Programmed Wait States) with PTADR#

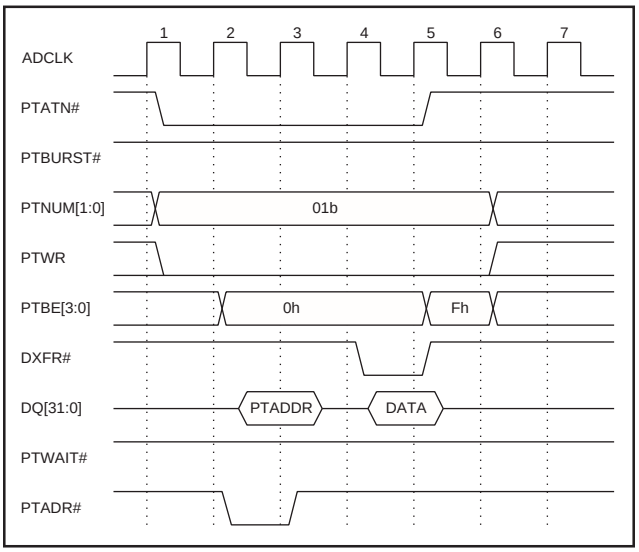
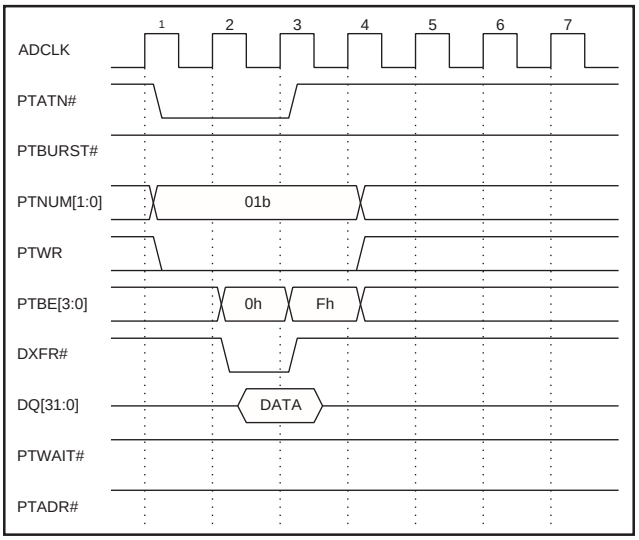


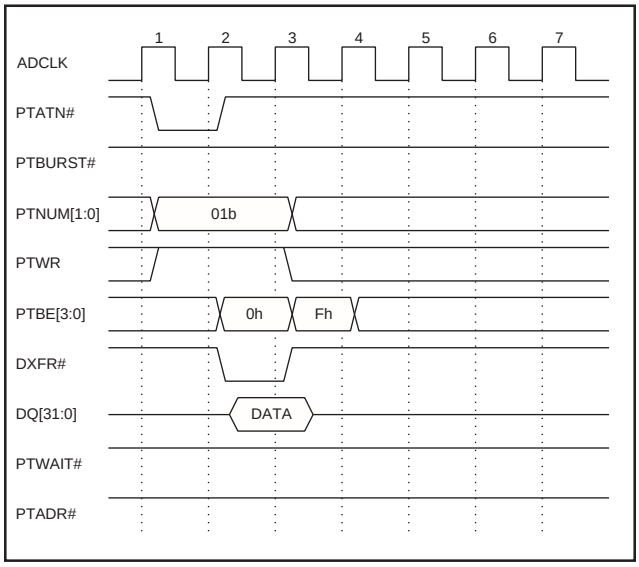
Figure 13b. Active Mode PCI Read without PTADR#



PTADR#

The PTADR# signal is controlled by the most significant bit of every region control field in the PTCR register. If this bit is zero then the PTADR# pin is not driven at the start of an Active mode transfer. If this bit is set to one, the PTADR# pin will be enabled and driven active (low) for one and only one clock after PTATN# was sampled active provided PTWAIT# was also sampled high.

Figure 13c. Active Mode PCI Write without PTADR#



When PTADR# is active (low), the S5920 will drive the DQ[31:0] bus with the 32-bit PCI address regardless of the PTMODE pin. To avoid contention on the DQ[31:0] bus during PCI read cycles, the S5920 incorporates a turnaround cycle before starting to drive the data (DXFR# assertion). This is needed only when PTADR# is enabled and when zero wait states are programmed during a Pass-Thru read cycle. The cycle immediately following the address cycle will be a turnaround cycle as shown in Figure 13a.

If PTADR# is disabled, the DXFR# output will be driven one clock cycle after PTATN# is valid (PTATN# is not considered active until PTATN# is low and PTWAIT# is high) regardless of the transfer being a read or a write. Figure 13b shows a PCI read cycle with PTADR# disabled.

Figure 13c shows a Pass-Thru write cycle with PTADR# disabled.

Active mode Programmable Wait States

Bits 0,1,2 of the PTCR register control this feature. Wait States are programmed on a per region basis. For example: region one can be set for zero wait states while other regions may have multiple wait states programmed.

Wait state options are 0,1,2,...7 wait states. The S5920 will always count N wait states (N=0,1,..7) before completing the current data phase.

Figures 17, 18 and 19 show Pass-Thru transfers with programmed wait states.

PTRDY#/PTWAIT#

In Active mode, the PTRDY#/PTWAIT# pin takes the PTWAIT# function, which is the opposite function of this pin when configured for passive mode. That is, if the part is configured to operate in Active mode, PTWAIT# asserted low means the Add-On wishes to insert wait states.

Add-On peripherals are allowed to insert wait state cycles at any time during an Active mode transfer. When PTWAIT# has been sampled low, the S5920 will tri-state its DQ[31:0] bus in order to allow other Add-On devices to use the bus without contention.

Figure 14. Active Mode PCI Write with Add-On Initiated Wait States Using PTWAIT#

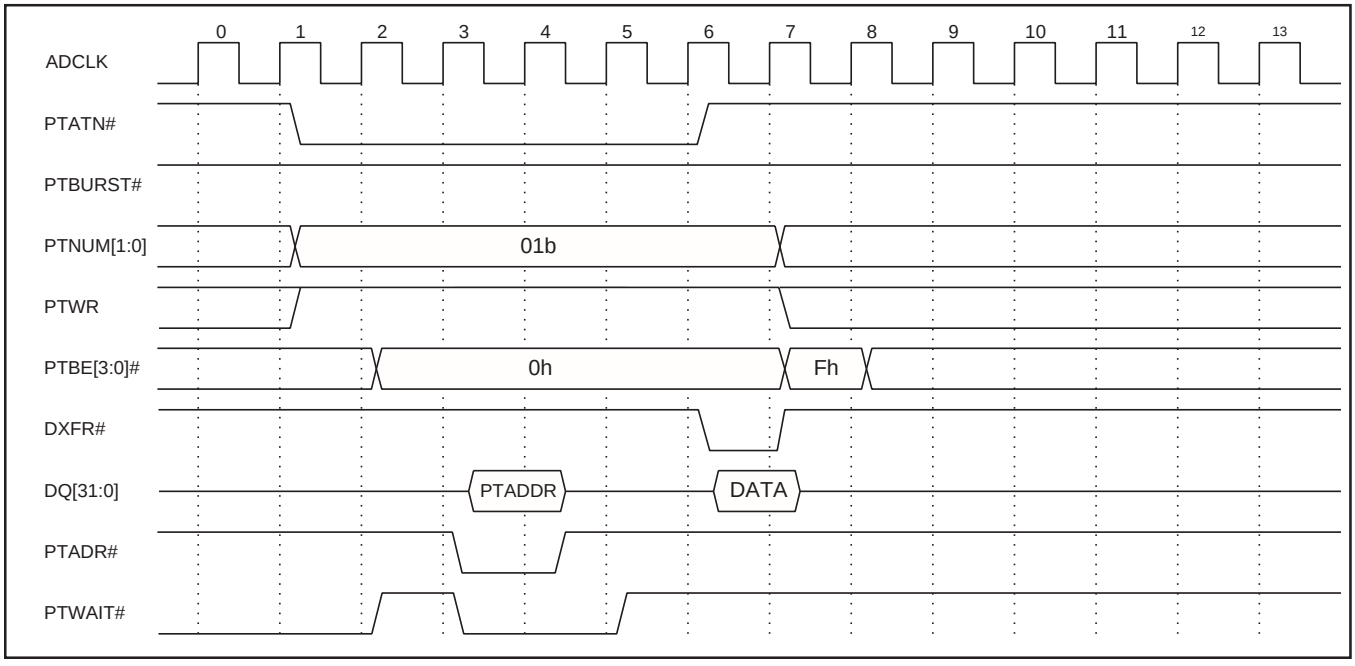
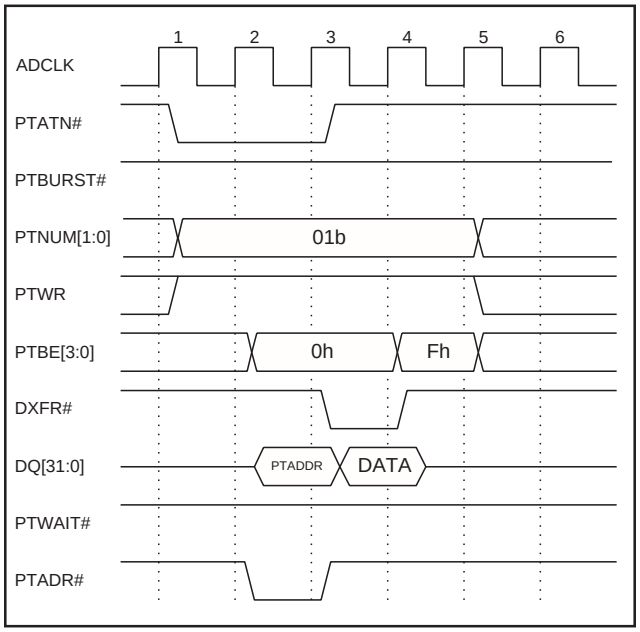


Figure 15. Active Mode 32-Bit PCI Write



The address phase of a Pass-Thru consists of the cycles from PTATN# asserted through PTADR# asserted (if PTADR# has been programmed to be disabled, there is no address phase). If PTWAIT# is asserted before the address phase, the address phase is delayed. The address phase will occur during the cycle after the clock edge that PTWAIT# is sampled high and PTATN# is sampled low,

The data phase(s) of a Pass-Thru consists of all the cycles after the (possibly nonexistent) address phase. During data phases, a wait is incurred during the cycle after PTWAIT# is sampled asserted.

Note: If PTWAIT# is activated in order to access other registers internal to the S5920, the user is responsible for inserting any needed turnaround cycles in order to avoid bus contention on the DQ bus.

DXFR#

DXFR# is a signal that is active during the cycles that a data transfer may take place. It is intended to be used to control strobes (e.g., write enable, read enable), and can be a flag for incrementing to the next address during a burst.

If wait states have been programmed, DXFR# will not go active until after all wait states have been executed. Note that asserting PTWAIT# to insert Add-On initiated wait states causes temporary suspension of the internal programmed wait state counter.

Active Mode Figures and Descriptions

Figure 14 shows a programmed zero wait state transfer in which the cycle start and the cycle completion are delayed by an external device controlling PTWAIT#.

Clock 1: The S5920 drives PTATN# active (low), indicating the start of a PCI to Add-On data transfer. PTBE[3:0] and PTNUM[1:0] are driven to their appropriate values for this transfer. PTWR is driven high indicating a PCI write.

Clock 2: This is a wait state since PTWAIT# was active (low) at the rising edge of clock 2.

Clock 3: PTWAIT# was inactive (high) at the rising edge of clock 3 so this cycle is the address phase: PTADR# is driven active (low) and the address value for the current transaction is driven onto the DQ bus.

Clock 4: PTADR# was active (low) at the rising edge of this clock so the Add-On device must latch the PCI address on the rising edge of this clock. The S5920 treats this cycle as a wait state since PTWAIT# was active (low) at the rising edge of clock 4.

Clock 5: This is a wait state since PTWAIT# was active (low) at the rising edge of clock 5.

Clock 6: PTWAIT# was inactive (high) at the rising edge of clock 6, so DXFR# is driven active (low) indicating a data transfer. PTATN# is driven inactive (high) indicating the Pass-Thru access is complete.

Clock 7: DXFR# was active (low) at the rising edge of this clock so the Add-On device must latch the PCI data on the rising edge of this clock. PTBE# is driven to Fh indicating all 4 bytes have been accessed. PTNUM and PTWR may change state since the Pass-Thru access is complete.

Clock 8: PTBE# may change state.

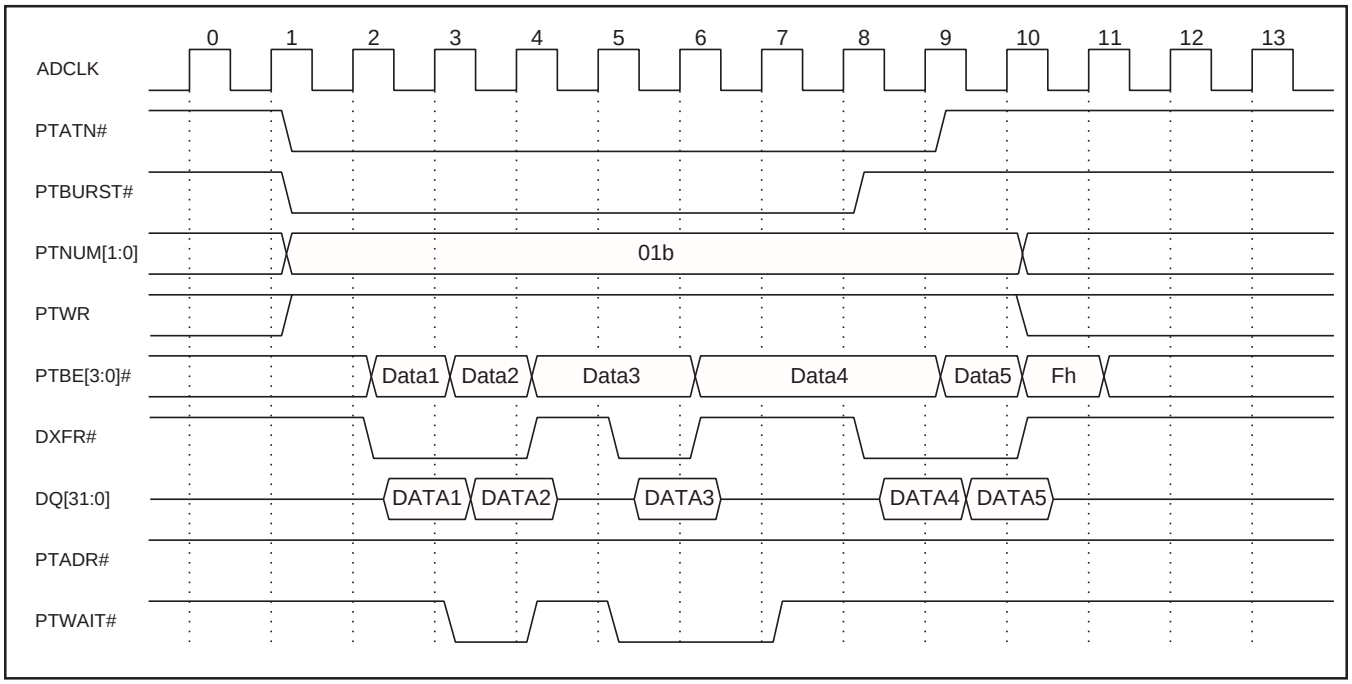
Figure 15 shows a single data phase 32-bit Active mode PCI Write with PTADR# enabled.

Active mode Burst cycles

PTBURST# signifies to the Add-On device that the current transfer will be contain more than one data phase. The Add-On device detects the end of a burst when the S5920 deasserts PTBURST#. During an Active mode PCI burst read, PTBURST# is deasserted when there is one more data word left to transfer. During an Active mode PCI burst write, PTBURST# deasserted indicates that after the current data word is transferred, there will be one data word left to transfer.

Figure 16 shows an Active mode PCI Burst Write with 0 programmed wait states. The Add-On device controlling PTWAIT# asserts wait states in the figure on an as-needed basis. PTADR# has been programmed to be disabled.

Figure 16. Active Mode 32-Bit PCI Write w/PTWAIT#



Clock by Clock description of Figure 16

Clock 1: The S5920 drives PTATN# and PTBURST# active (low), indicating the start of a PCI to Add-On data transfer with more than one data cycle. PTBE[3:0] and PTNUM[1:0] are driven to their appropriate values for this transfer. PTWR is driven high indicating a Pass-Thru write.

Clock 2: Since this region does not have PTADR# enabled as an output and PTWAIT# is high at the rising edge of clock 2, the first data transfer is indicated by driving DXFR# low and the data on the data bus DQ[31:0] .

Clock 3: DXFR# is sampled active by the Add-On device which indicates that the Add-On device must latch the first data word at the rising edge of this clock. Valid data is determined by decoding the PTBE[3:0]# lines. The Add-On device drives PTWAIT# active (low) requesting a wait state on the next cycle.

Clock 4: DXFR# is sampled active (low) by the Add-On device which indicates that the Add-On device must latch the second data word at this clock edge. The S5920 tri-states its output bus since PTWAIT# was inactive (high) at the rising edge of clock 4. Additionally, the S5920 deasserts DXFR# indicating that no data transfer will occur on the next clock edge (this is because this cycle is a wait state since PTWAIT# was active (low) at the rising edge of clock 4. The Add-On device deasserts PTWAIT# indicating no wait state on the next clock.

Clock 5: No data transfer takes place at the rising edge of clock 5 since the previous cycle was an Add-On initiated wait state (because PTWAIT# was active (low) at the rising edge of clock 4). The S5920 asserts DXFR# and drives the third data onto the DQ bus since PTWAIT# was inactive (high) at the rising edge of clock 5. The Add-On device drives PTWAIT# active (low) requesting a wait state on the next cycle.

Clock 6: DXFR# is sampled active by the Add-On device which indicates that the Add-On device must latch the third data word at the rising edge of this clock. The S5920 drives DXFR# inactive and tri-states the DQ bus since PTWAIT# was active (low) at the rising edge of clock 6. The Add-On keeps PTWAIT# asserted indicating it wants to add a wait state on the next cycle.

Clock 7: No data transfer takes place on the rising edge of this clock since the previous cycle was an Add-On initiated wait state (because PTWAIT# was active (low) at the rising edge of clock 6).

Clock 8: No data transfer takes place on the rising edge of this clock since the previous cycle was an Add-On-initiated wait state (because PTWAIT# was active (low) at the rising edge of clock 7). DXFR# is driven active (low) and the fourth data is driven onto the DQ bus since PTWAIT# was inactive (high) at the rising edge of clock 8. PTBURST# is driven inactive (high) indicating that after this data word is transferred, there is only one data word left to transfer.

Clock 9: DXFR# is sampled active (low) by the Add-On device which indicates that the Add-On device must latch the fourth data word at the rising edge of this clock. PTATN# is driven inactive (high) indicating that this will be the last data phase.

Clock 10: DXFR# is sampled active (low) by the Add-On device which indicates that the Add-On device must latch the fifth data word at the rising edge of this clock. DXFR# is deasserted since the access is complete. PTBE# is driven to Fh indicating all 4 bytes have been accessed. PTNUM and PTWR may change state since the access is complete.

Clock 11: PTBE# may change state.

Figure 17 shows a Pass-Thru Burst Write data transfer in which the S5920 has been programmed to strobe data using a one-wait state delay. The Add-On device leaves PTWAIT# inactive (high) for all time.

Clock by Clock description of Figure 17

Clock 1: The S5920 drives PTATN# and PTBURST# active (low) indicating the start of a PCI to Add-On data transfer with more than one data cycle. PTBE[3:0] and PTNUM[1:0] are driven to their appropriate values for this transfer. PTWR is driven high indicating a Pass-Thru write.

Figure 17. Active Mode PCI Write Showing a One Wait State Programmed Delay

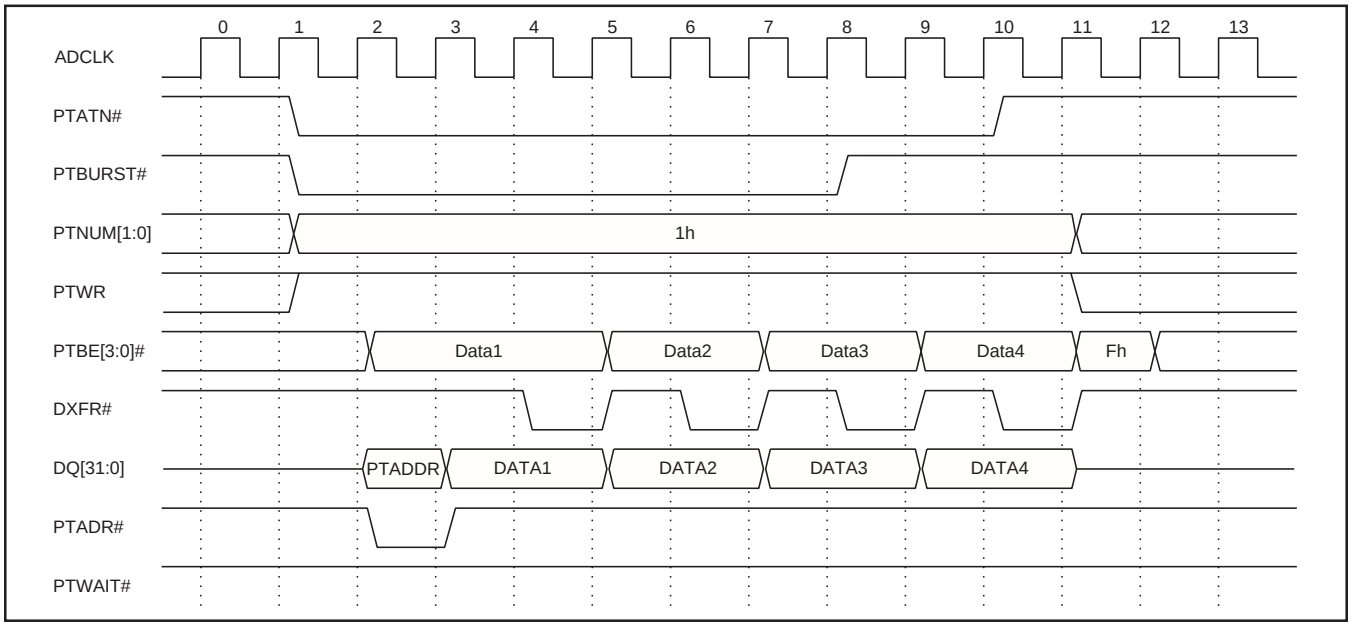
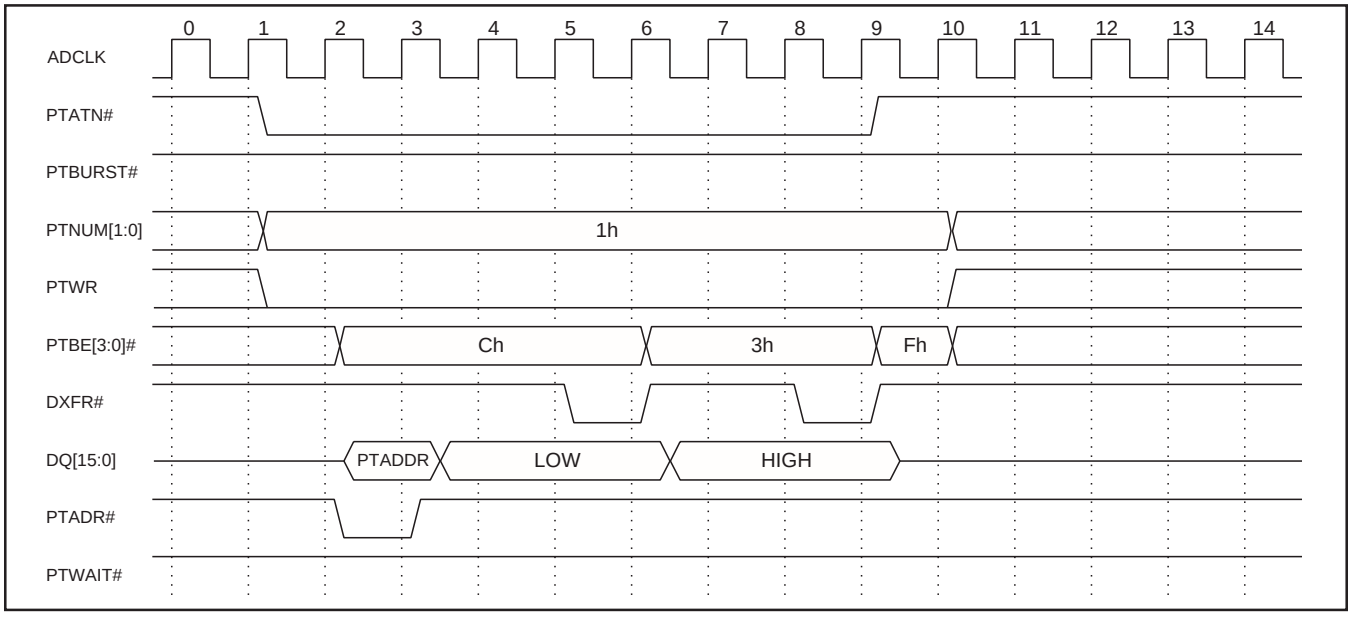


Figure 18. 16-Bit Active Mode PCI Read w/ Programmed Wait States



Clock 2: Since this region does have PTADR# enabled as an output, it is driven active (low) and the PCI address for the current transaction is presented on the DQ[31:0] bus.

Clock 3: The Add-On device must latch the PCI address at the rising edge of this clock.

Clock 4: DXFR# is asserted low indicating that data will be transferred on the next rising clock edge (clock 5). Data1 is driven onto the DQ[31:0] bus.

Clock 5: The Add-On device must latch the first data word at the rising edge of this clock. Valid data is determined by decoding the PTBE[3:0]# lines.

Clock 6: DXFR# is asserted indicating that data will be transferred on the next rising clock edge (clock 7). DATA2 is driven onto the DQ[31:0] bus.

Clock 7: The Add-On side device latches the second data word (DATA2) at the rising edge of this clock.

Clock 8: DXFR# is asserted indicating that data will be transferred on the next rising clock edge (clock 9). DATA3 is driven onto the DQ[31:0] bus. PTBURST# is driven inactive indicating that after this data word is transferred, there is only one data word left to transfer.

Clock 9: The Add-On side device latches the third data word (DATA3) at this clock edge.

Clock 10: DXFR# is asserted indicating that data will be transferred on the next rising clock edge (clock 11). DATA4 is driven onto the DQ[31:0] bus. PTATN# is deasserted indicating that this will be the last data phase.

Clock 11: The final data word (DATA4) must be latched by the Add-On device at the rising edge of this clock. PTBE# is driven to Fh indicating all 4 bytes have been accessed. PTNUM and PTWR may change state since the access is complete.

Clock 12: PTBE# may change state.

Active Mode with 16/8-bit data buses

When the S5920 is programmed in Active mode and 16-bit, the DXFR# output will strobe twice for every PCI 32-bit word that has been read/written. Each DXFR# assertion signifies that a 16-bit word has been transferred to the Add-On side. The first DXFR# completion will be for the least significant 16-bit word of a 32-bit word ("LOW" in Figures 18, 9-20 and 9-21), while the second transfer will be for the most significant 16-bit word ("HIGH" in Figures 18, 9-20 and 9-21). If the current PCI access has only 2 bytes valid (PCI BE[3:0]# encoding of Ch or 3h instead of 0h), then the S5920 will still assert a 2 cycle completion but one of them will not contain valid data (PTBE[3:0]#=Fh). If the programmed wait states for the current Pass-Thru region is not zero, then the S5920 will insert the programmed wait states before the "LOW" data word and also between the "LOW" and "HIGH" data words. Figure 18 shows a PCI read to a 16-bit Add-On region with two programmed wait states. Note that a PCI read to an 8-bit Add-On would be the same as Figure 18 except that there would be 4 data transfers (one for each byte) vice 2.

As in Passive mode, in Active mode, the word read/write order is determined by the Endian conversion programmed into the S5920.

Figure 19. Active Mode PCI Read w/ Programmed Wait States

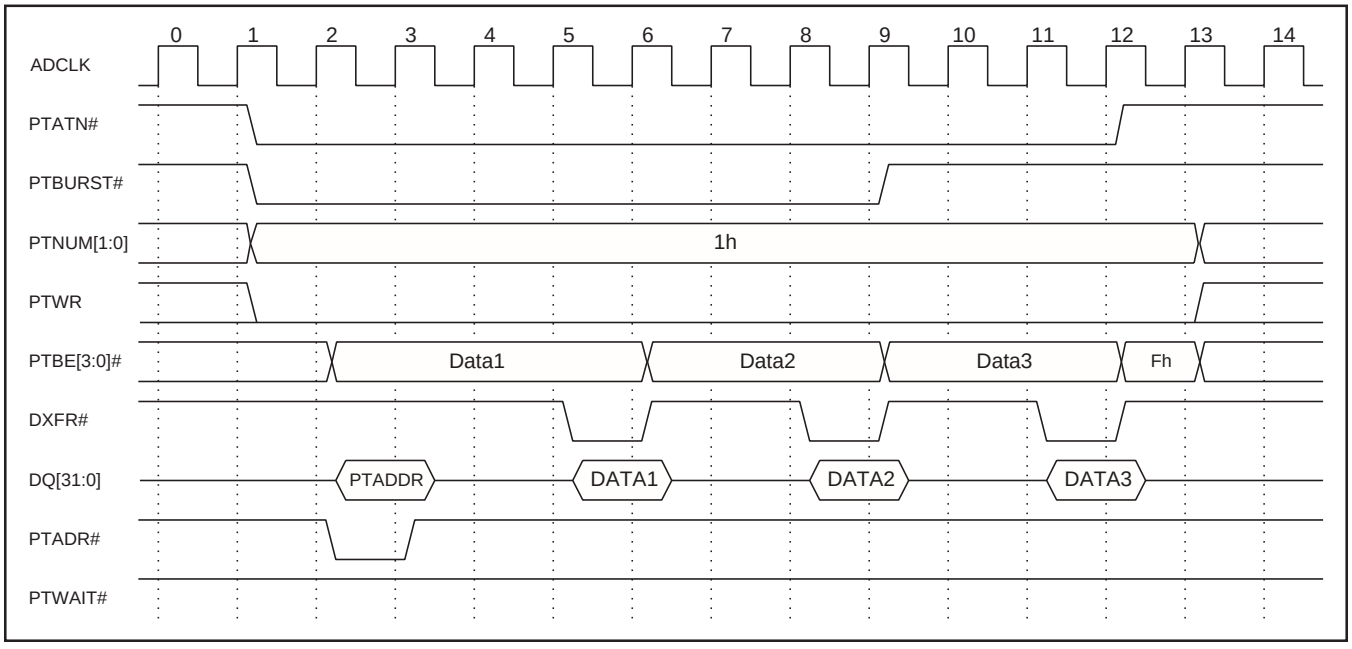


Figure 20. Active Mode PCI Read

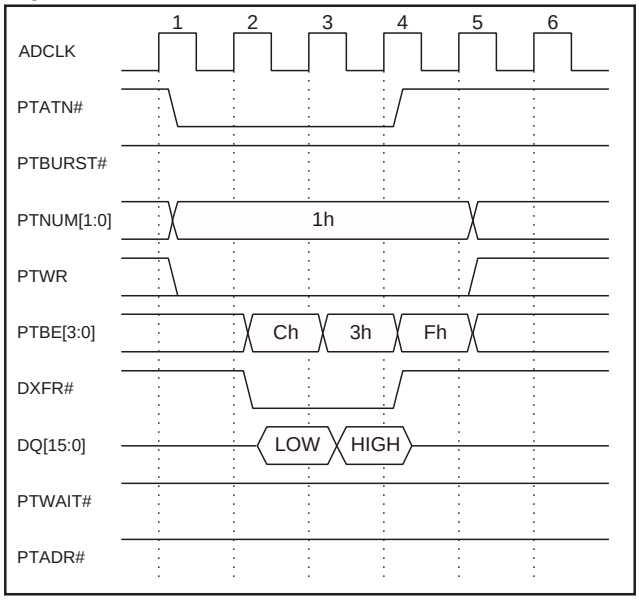


Figure 21. Active Mode PCI Write

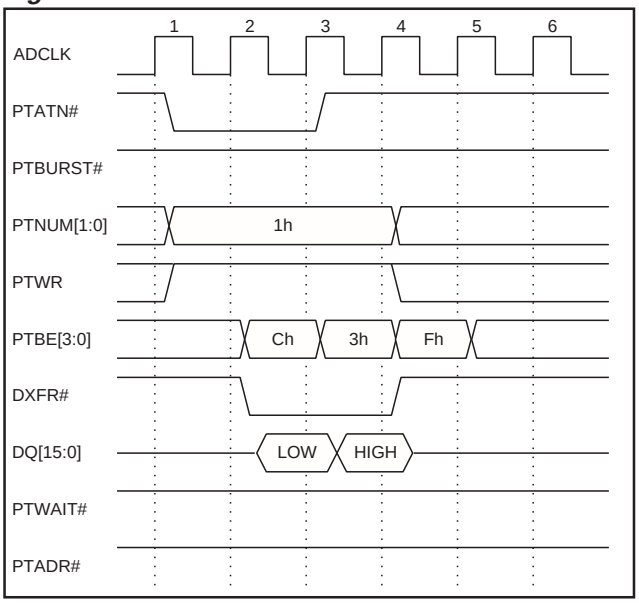


Figure 20 shows a Pass-Thru write cycle with 0 wait states for a 16-bit region. The PTBE[3:0]# signals are used by the Add-On device to determine validity of the current word cycle and also, which word of a long word is currently being driven by the S5920. PTBE[3:0]# encoding of Ch indicates the least significant 16-bit portion of the 32-bit PCI data word is on the DQ[15:0] bus. PTBE[3:0]# encoding of 3h indicates the most significant 16-bit portion of the 32-bit PCI data word is on the DQ[15:0] bus. PTADR# is shown as disabled in Figure 20.

Figure 21 shows a Pass-Thru read cycle with 0 wait states for a 16-bit region. PTADR# is disabled.

If the Add-On bus size is 8 bits, then the S5920 will assert DXFR# 4 times for each 32-bit PCI word The first completion is for byte 0, the second is for byte 1, the third is for byte 3, and the fourth DXFR# assertion is for byte 4 of a 32-bit word. If the current PCI access has less than four bytes valid (PCI BE[3:0]# encoding is not 0h), then the S5920 will still assert a 4-cycle completion but one or more of them will not contain valid data (PTBE[3:0]# = Fh).

As in Passive mode, in Active mode, the word read/ write order is determined by the Endian conversion programmed into the S5920.

Figure 22 shows a Pass-Thru write cycle with 0 wait states for an 8-bit region. The PTBE[3:0]# signals are used by the Add-On device to determine validity of the current byte cycle and also, which byte of a long word is currently being driven by the S5920. PTADR# is enabled as an output.

CONFIGURATION

The S5920 Pass-Thru interface utilizes four Base Address Registers (BADR1:4). Each Base Address Register corresponds to a Pass-Thru region. The contents of these registers during initialization determine the characteristics of that particular Pass-Thru region. Each region can be mapped to memory or I/O space. Memory mapped devices can, optionally, be mapped below 1 Mbyte and can be identified as prefetchable. Both memory and I/O regions can be configured as 8, 16 or 32 bits wide.

Base Address Registers are loaded during initialization from the external non-volatile boot device. Without an external boot device, the default value for the BADR registers is zero (region disabled). The Base Address Registers are the only registers that define Pass-Thru operation. Consequently, the Pass-Thru interface cannot be used without an external non-volatile boot device.

S5920 Base Address Register Definition

Certain bits in the Base Address Register have specific functions:

D0 Memory or I/O mapping. If this bit is clear, the region should be memory mapped. If this bit is set, the region should be I/O mapped.

D2:1 Location of a memory region. These bits request that the region be mapped in a particular part of memory. These bit definitions are only used for memory mapped regions.

D2	D1	Location
0	0	Anywhere in 32-bit memory space
0	1	Below 1 Mbyte in memory space (Real Mode address space)
1	0	Anywhere in 64-bit memory space (not valid for the S5920)
1	1	Reserved

D3 Prefetchable. For memory mapped regions, the region can be defined as cacheable. If set, the region is cacheable. If this bit is clear, the region is not.

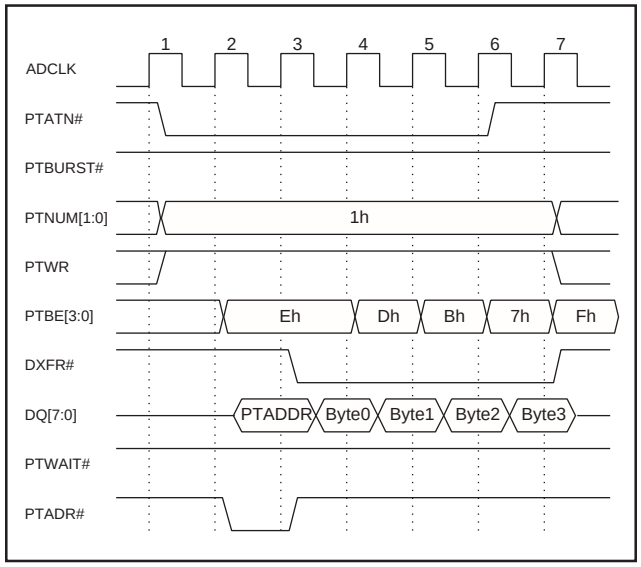
D31:30 Pass-Thru region bus width. These two bits are used by the S5920 to define the data bus width for a Pass-Thru region. Regardless of the programming of other bits in the BADR register, if D31:30 are zeros, the Pass-Thru region is disabled.

D31	D30	Add-On Bus Width
0	0	Region disabled
0	1	8 bits
1	0	16 bits
1	1	32 bits

BADR1:4 bits D31:30 are used only by the S5920. When the host reads the Base Address Registers during configuration cycles, they always return the same value as D29. If D29 is zero, D31:30 return zero, indicating the region is disabled. If D29 is one, D[31:30] return one. This operation limits each Pass-Thru region to a maximum size of 512 Mbytes of memory.

For I/O mapped regions, the PCI specification allows no more than 256 bytes per region. The S5920 allows larger regions to be requested by the Add-On, but a PCI BIOS will not allocate the I/O space and will probably disable the region.

Figure 22. 8-Bit Active Mode PCI Write



Creating a Pass-Thru Region

Section 3.11 describes the values that must be programmed into the non-volatile boot device to request various block sizes and characteristics for Pass-Thru regions. After reset, the S5920 downloads the contents of the boot device locations 54h, 58h, 5Ch, and 60h into "masks" for the corresponding Base Address Registers. The following are some examples for various Pass-Thru region definitions:

NV Memory Contents	Pass-ThruRegion Definition
54h = BFFFF002h	Pass-Thru region 1 is a 4 Kbyte region, mapped below 1 Mbyte in memory space with a 16-bit Add-On data bus. This memory region is not cacheable.
58h = 3xxxxxxxh	Pass-Thru region 2 is disabled. (D31:30 = 00.)
5Ch = FFFFFF81h	Pass-Thru region 3 is a 32-bit, 128 byte I/O-mapped region.
60h = 00000000h	Pass-Thru region 4 is disabled.

During the PCI bus configuration, the host CPU writes all ones to each Base Address Register, and then reads the contents of the registers back. The mask downloaded from the boot device determines which bits are read back as zeros and which are read back as ones. The number of zeros read back indicates the amount of memory or I/O space a particular S5920 Pass-Thru region is requesting.

After the host reads all Base Address Registers in the system (as every PCI device implements from one to six), the PCI BIOS allocates memory and I/O space to each Base Address region. The host then writes the start address of each region back into the Base Address Registers. The start address of a region is always an integer multiple of the region size. For example, a 64-Kbyte memory region is always mapped to begin on a 64K boundary in memory. It is important to note that no PCI device can be absolutely located in system memory or I/O space. All mapping is determined by the system, not the application.

PCI or Add-On Operation registers PTCR or APTCR provide additional configuration control for each region.

Accessing a Pass-Thru Region

After the system is finished defining all Base Address Regions within a system, each Base Address Register contains a physical address. The application software must now find the location in memory or I/O space of its hardware. PCI systems provide BIOS or operating system function calls for application software to find particular devices on the PCI bus based on Vendor ID and Device ID values. This allows application software to access the device's Configuration Registers.

The Base Address Register values in the S5920's Configuration Space may then be read and stored for use by the program to access application hardware. The value in the Base Address Registers is the physical address of the first location of that Pass-Thru region. Some processor architectures allow this address to be used directly to access the PCI device. For Intel Architecture systems, the physical address must be changed into a Segment/Offset combination.

For Real Mode operation in an Intel Architecture system (device mapped below 1 Mbyte in memory), creating a Segment/Offset pair is relatively simple. To calculate a physical address, the CPU shifts the segment register 4 bits to the left and adds the offset (resulting in a 20 bit physical address). The value in the Base Address Register must be read and shifted 4 bits to the right. This is the segment value and should be stored in one of the Segment registers. An offset of zero (stored in SI, DI or another offset register) accesses the first location in the Pass-Thru region.

Special Programming Features

A few additional features have been provided to the user which will allow for optimal "tuning" of their system. As these are not features that will be changed "on the fly", they have been included as part of the nvRAM boot-up sequence. nvRAM address 45h is a memory location in the external nvRAM which will contain these custom programmed bits. These features, and their corresponding bit at location 45h, are described as follows:

LOC_45(h)	Description	Default
b = 0	Target Latency Enb	1
b = 1	Retry Flush Enb	0
b = 2	Write FIFO Mode	0
b = (7:3)	Reserved	X

Target Latency describes the number of cycles that a target device may respond to a PCI data transfer request. The PCI 2.1 specification indicates that the target device has 16 clocks to respond to an initial request (from the assertion of FRAME#), and 8 clocks from each subsequent data phase. If the target is not capable of asserting TRDY# within these time frames, it must assert a STOP#, thus initiating a disconnect. The Target Latency programmed bit allows the user to disable the generation of disconnects in the event of a slow Add-On device.

If Target Latency Enb is low, target latency is ignored. In this case, the S5920 will never issue a retry/disconnect in the event of a slow add-on device. Instead, TRDY# wait states will be asserted. This might be useful for an embedded system, where the S5920 can take up as many clock cycles as necessary to complete a transfer. This programmable bit is only provided for flexibility and most users should leave this bit set to 1. If Target Latency Enb is high, the device will be PCI 2.1 compliant with respect to Target Latency.

Retry Flush Enb indicates to the Pass-Thru whether to hold prefetched data following a disconnect, or to allow the data to be flushed out during the next PCI read access. If low, the data will be held in the PT read FIFO until the initiator comes back to read it out. All subsequent PCI accesses to the S5920 from a device other than the one who initiated the read will be acknowledged with a retry. If the master never returns for the data, the Pass-Thru function will be hung. Even though the PCI 2.1 Specification does not require a master to return for data following a disconnect, it is unlikely that a master will terminate a read transfer until all data has been collected.

If Retry Flush Enb is high, the data will be flushed from the FIFO if a subsequent PCI read access is not to the same address. If the original master received a retry (disconnect, but with no data transfer), the read data is held in the FIFO until the master comes back for it. In this case, the Retry Flush Enb has no effect. The PCI 2.1 Specification states that the master must come back if it receives a retry.

Write FIFO Mode indicates what to do in the event that a full FIFO is detected during a PCI write transfer. If low, the S5920 will perform an immediate disconnect, thus freeing up the PCI bus for other transfers. The initiator will have to come back to complete the data transfer, after which time the FIFO should no longer be full. If Write FIFO Mode is high, the S5920 will deassert TRDY#, and allow for either another FIFO location to become available (as the Add-On has read a DWORD), or wait for the Target Latency to expire (8 clocks from previous data phase), thus initiating a disconnect. This will allow for the Add-on device to “catch-up” without losing the burst.

ABSOLUTE MAXIMUM STRESS RATINGS

Table 1 lists the absolute maximum S5920 device stress ratings. Stresses beyond those listed may cause permanent damage to the device. These are stress ratings only; operation of the device at these or any other conditions beyond those indicated in the Operating Characteristics section of this specification is not implied.

Table 1. Absolute Maximum Stress Ratings

Parameter	Min	Max	Units
Storage Temperature	-55	125	^{0}C
Supply Voltage (V_{DD})	- 0.3	7.0	Volts
Input Pin Voltage	- 0.5	GND + 5.0	Volts

S5920 Operating Conditions

Table 2 lists the S5920 operating conditions. These parameters are guaranteed by device characterization, but not device tested. All values are maximum guaranteed values.

Table 2. Operating Conditions

Symbol	Parameter	Min	Max	Units	Test Conditions	Notes
V_{cc}	Supply Voltage	4.75	5.25	Volts		
I_{cc}	Supply Current (Static)	-	49	mA	V_{CC} = 5.25 Volts	
I_{cc}	Supply Current (Dynamic)	-	197	mA	V_{CC} = 5.25 Volts	
T_A	Operating Temperature	0	70	^{0}C		
0_{JA}	Thermal Resistance	-	TBD	^{0}C/W		
P_{DD}	Power Dissipation	0	TBD	W	V_{CC} = 5.25 Volts	

PCI Signal DC Characteristics

The following table summarizes the DC characteristic parameters for all PCI signals listed below as they apply to the S5920.

AD[31:0] (t/s), PAR (t/s), C/BE[3:0]# (in), FRAME# (in), IRDY# (in), TRDY# (s/t/s), STOP# (s/t/s), LOCK# (in), IDSEL (in), DEVSEL# (s/t/s), PERR# (s/t/s), SERR# (o/d), INTA# (o/d), RST# (in), CLK (in).

Table 3. PCI Signal DC Characteristics *(V_{CC} = 5.0V 5%, 0^0C to 70 ^{0}C, 50 pF load on outputs)*

Symbol	Parameter	Min	Max	Units	Test Conditions	Notes
V_{ih}	Input High Voltage	2.0	-	Volts		1
V_{il}	Input Low Voltage	-0.5	0.8	Volts		1
I_{ih}	Input High Leakage Current	-	70	uA	V_{IN} = 2.7	2
I_{il}	Input Low Leakage Current	-	-70	uA	V_{IN} = 0.5	2
V_{oh}	Output High Voltage	2.4	-	Volts	I_{OUT} = -2mA	
V_{ol}	Output Low Voltage	-	0.55	Volts	I_{OUT} = 3mA, 6mA	3
C_{in}	Input Pin Capacitance	-	10	pF		4
C_{CLK}	CLK Pin Capacitance	5	12	pF		
C_{IDSEL}	IDSEL Pin Capacitance	-	8	pF		

Notes:
1. Recommended values for all PCI signals except CLK to be Vih = 2.4 Min and Vil = .4 Max.
2. Input leakage applies to all inputs and bi-directional buffers.
3. PCI bus signals without pull-up resistors will provide the 3 mA output current. Signals which require a pull-up resistor will provide 6 mA output current.
4. The PCI specification limits all PCI inputs not located on the motherboard to 10 pF (the clock is allowed to be 12 pF).

Add-On Signal DC Characteristics

The following table summarizes the Add-On DC characteristic parameters for the Add-On signals listed below as they apply to the S5920. All Add-On signal outputs listed are capable of sinking or sourcing 4 mA with the exception of BPCLK which can sink or source 8 mA.

DQ[31:0] (t/s), ADR[6:2] (in), BE[3:0]# (in), SELECT# (in), RD# (in), WR# (in), PTATN# (out), PTBE[3:0]# (out), PTNUM[1:0] (out), PTWR (out), PTBURST# (out), PTADR# (I/O), PTRDY# (in), PTMODE (in), DXFR# (out), SYSRST# (out), IRQ# (out), ADDINT# (in), BPCLK (out), ADCLK (in), DQMODE (in), MDMODE (in), MD[7:0] (I/O), LOAD# (in).

Table 4. Add-On Operating Characteristics (V_{CC} *= 5.0V 5%, 0°C to 70 °C, 50 pF load on outputs)*

Symbol	Parameter	Min	Max	Units	Test Conditions	Notes
V_{ih}	Input High Voltage	2.0	-	Volts		
V_{il}	Input Low Voltage	-0.5	0.8	Volts		
I_{ih}	Input High Leakage Current	-10	+10	uA	V_{IN} = 2.7	
I_{il}	Input Low Leakage Current	-10	+10	uA	V_{IN} = 0.5	
V_{oh}	Output High Voltage	3.5	-	Volts		
V_{ol}	Output Low Voltage	-	0.4	Volts		

nvRAM Memory Interface Signals

SCL 8 mA (o/d)

SDA 8 mA (bi-directional-o/d)

TIMING SPECIFICATION

PCI Clock Specification

Table 5 summarizes the A. C. characteristics for the PCI bus signals as they apply to the S5920. The figures after Table 5 visually indicate the timing relationships.

Table 5. PCI Clock Specifications

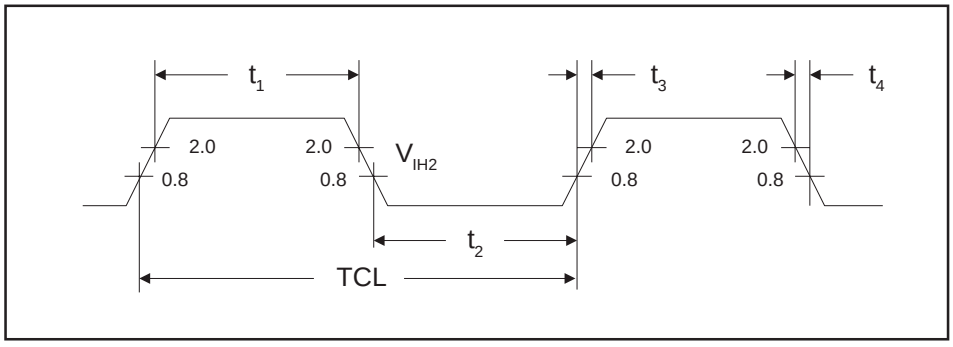
Functional Operation Range (V_{CC} = 5.0V +/- 5%, 0°C to 70°C, 50 pF load on outputs for MAX, 0 pF load for MIN).

Symbol	Parameter	Min	Max	Units	Notes
T_{cyc}	Clock Time	30		ns	
t_1	CLK High Time	11		ns	
t_2	CLK Low Time	11		ns	
t_3	Rise Time (0.8V to 2.0V)	1	4	V/ns	1
t_4	Fall Time (2.0V to 0.8V)	1	4	V/ns	1
t_5	CLK to Signal Valid Delay (Bused Signals) CLK to Signal Valid Delay (Point-to-Point Signals)	2 2	11 12	ns ns	1,2
t_6	Float to Active Delay	2		ns	3
t_7	Active to Float Delay		28	ns	3
t_8	Rising Edge Setup	7		ns	4
t_9	Hold from PCI Clock Rising Edge	0		ns	4

Notes:

1. Rise and fall times are specified in terms of the edge rate measured in V/ns. This slew rate is met across the minimum peak-to-peak portion of the clock waveform as shown in Figure 1.
2. Minimum times are evaluated with 0 pF equivalent load; maximum times are evaluated with 50 pF equivalent load.
3. For purposes of Active/Float timing measurements, the Hi-Z or 'off" state is defined to be when the total current delivered through the component pin is less than or equal to the leakage current specification.
4. See the timing measurement conditions in Figure 3.

Figure 1. PCI Clock Timing



Figures 2 and 3 define the conditions under which timing measurements are made. The user designs must guarantee that minimum timings are met with maximum clock skew rate (fastest edge) and voltage swing.

Figure 2. PCI Signal Output Timing

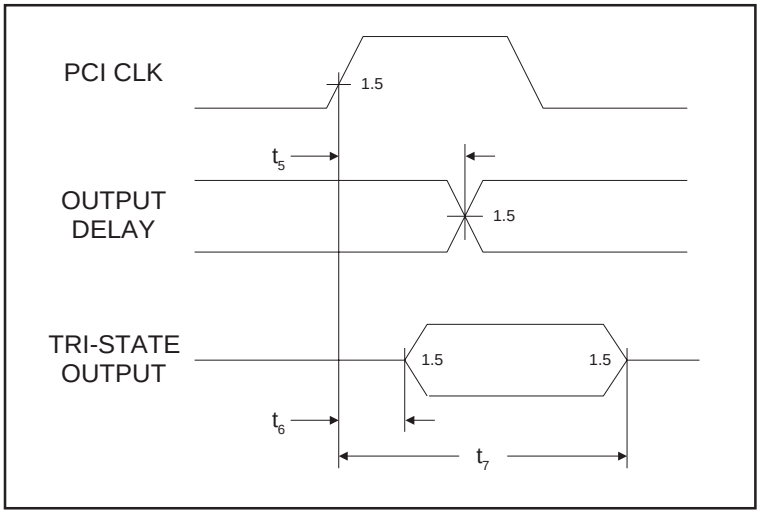
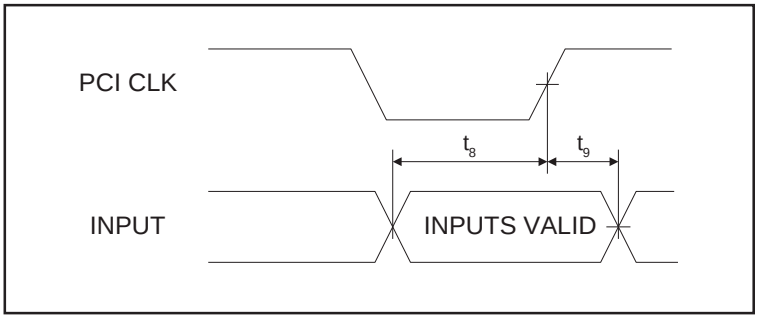


Figure 3. PCI Signal Input Timing



Add-On Signal Timings

Table 6 summarizes the A. C. characteristics for the Add-On bus signals as they apply to the S5920. The figures after Table 6 visually indicate the timing relationships.

Table 6. Add-On Timings

Functional Operation Range (V_{CC}= 5.0V +/- 5%, 0ºC to 7º0 C, 50 pF load on outputs for MAX, 0 pF load for MIN).

Symbol	Parameter	Min	Max	Units	Notes
TACL	ADCLK Cycle Time	25		ns	
t_{10}	ADCLK High Time	10		ns	
t_{11}	ADCLK Low Time	10		ns	
t_{12}	ADCLK Rise Time (0.8V to 2.0V)		2.5	ns	
t_{13}	ADCLK Fall Time (2.0V to 0.8V)		2.5	ns	
t_{14}	PCICLK to BPCLK Delay, Rising		5.3	ns	
t_{15}	PCICLK to BPCLK Delay, Falling		6.2	ns	
t_{16}	PTADR# Low to DQ[31:0] Output Valid		13.1	ns	1
t_{17a}	PTADR# High to DQ[31:0] Output Hold	2		ns	1
t_{17b}	PTADR# High to DQ[31:0] Output Float		11.9	ns	1
t_{18}	PTATN# Valid from ADCLK Rising Edge		13.5	ns	
t_{19}	PTATN# Hold from ADCLK Rising Edge	4		ns	
t_{20}	PTBURST# Valid from ADCLK Rising Edge		13	ns	
t_{21}	PTBURST# Hold from ADCLK Rising Edge	4		ns	
t_{22}	PTNUM[1:0] Valid from ADCLK Rising Edge		14.4	ns	
t_{23}	PTNUM[1:0] Hold from ADCLK Rising Edge	4		ns	
t_{24}	PTWR Valid from ADCLK Rising Edge		13.1	ns	
t_{25}	PTWR Hold from ADCLK Rising Edge	4		ns	
t_{26}	PTBE[3:0]# Valid from ADCLK Rising Edge		14.4	ns	
t_{27}	PTBE[3:0]#Hold from ADCLK Rising Edge	3		ns	
t_{28}	PTWAIT# Setup to ADCLK Rising Edge	11		ns	
t_{29}	PTWAIT# Hold from ADCLK Rising Edge	1		ns	
t_{30}	SELECT# Setup to ADCLK Rising Edge	8.9		ns	
t_{31}	SELECT# Hold from ADCLK Rising Edge	1		ns	
t_{32}	ADR[6:2] Setup to ADCLK Rising Edge	9.3		ns	
t_{33}	ADR[6:2] Hold from ADCLK Rising Edge	1		ns	
t_{34}	BE[3:0]# Setup to ADCLK Rising Edge	8.3		ns	
t_{35}	BE[3:0]# Hold from ADCLK Rising Edge	1		ns	
t_{36}	RD# Setup to ADCLK Rising Edge	6.7		ns	
t_{37}	RD# Hold from ADCLK Rising Edge	1		ns	
t_{38}	DQ[31:0] Output Valid from ADCLK Rising Edge		16	ns	2
t_{39}	DQ[31:0] Output Hold from ADCLK Rising Edge	3		ns	2
t_{40}	DQ[31:0] Output Float from ADCLK Rising Edge		15.6	ns	2
t_{41}	WR# Setup to ADCLK Rising Edge		7.4	ns	
t_{42}	WR# Hold from ADCLK Rising Edge	1		ns	
t_{43}	DQ[31:0] Input Setup to ADCLK Rising Edge	5.4		ns	
t_{44}	DQ[31:0] Input Hold from ADCLK Rising Edge	1		ns	

Notes:

1. Refers to Pass-Thru Passive mode only.
2. Refers to Pass-Thru Active and Passive modes.

Figure 4. Add-On Clock Timing

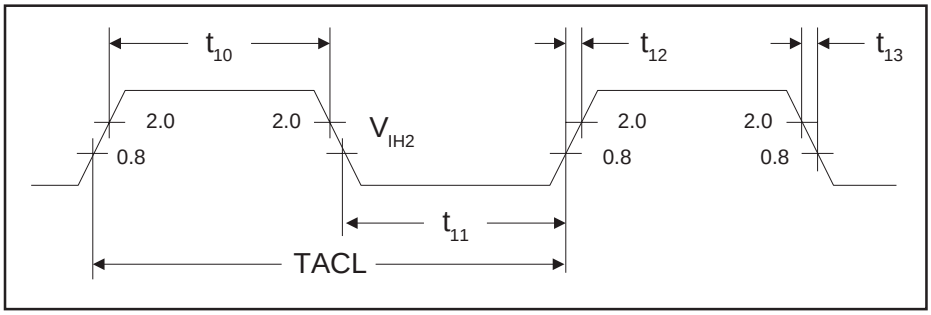


Figure 5. Pass-Thru Clock Relationship to PCI Clock

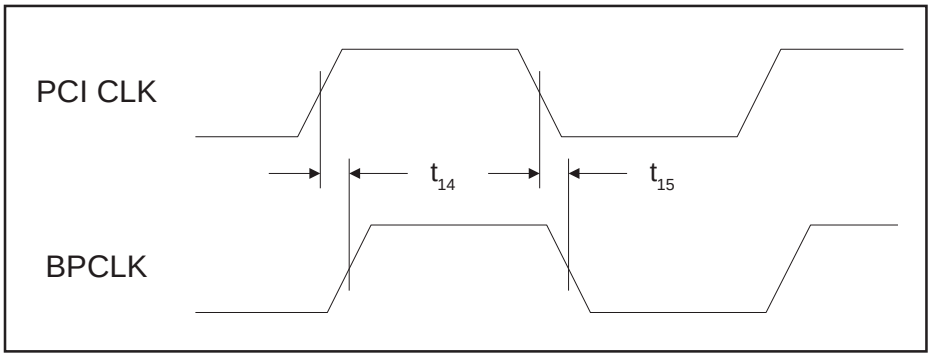


Figure 6. PTADR Timing

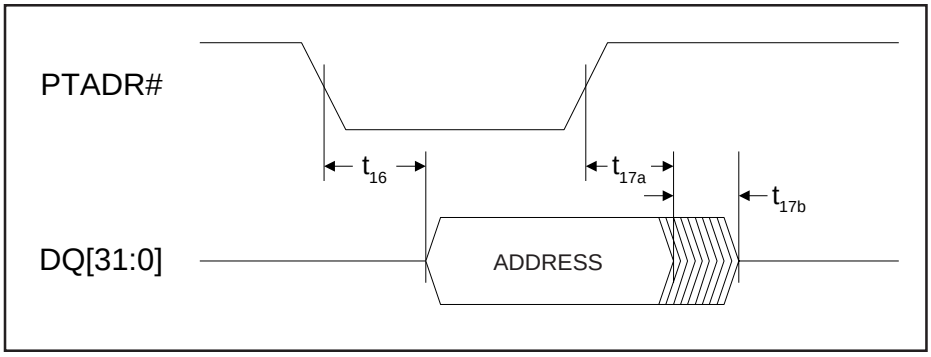


Figure 7. Passive Mode Pass-Thru Operation

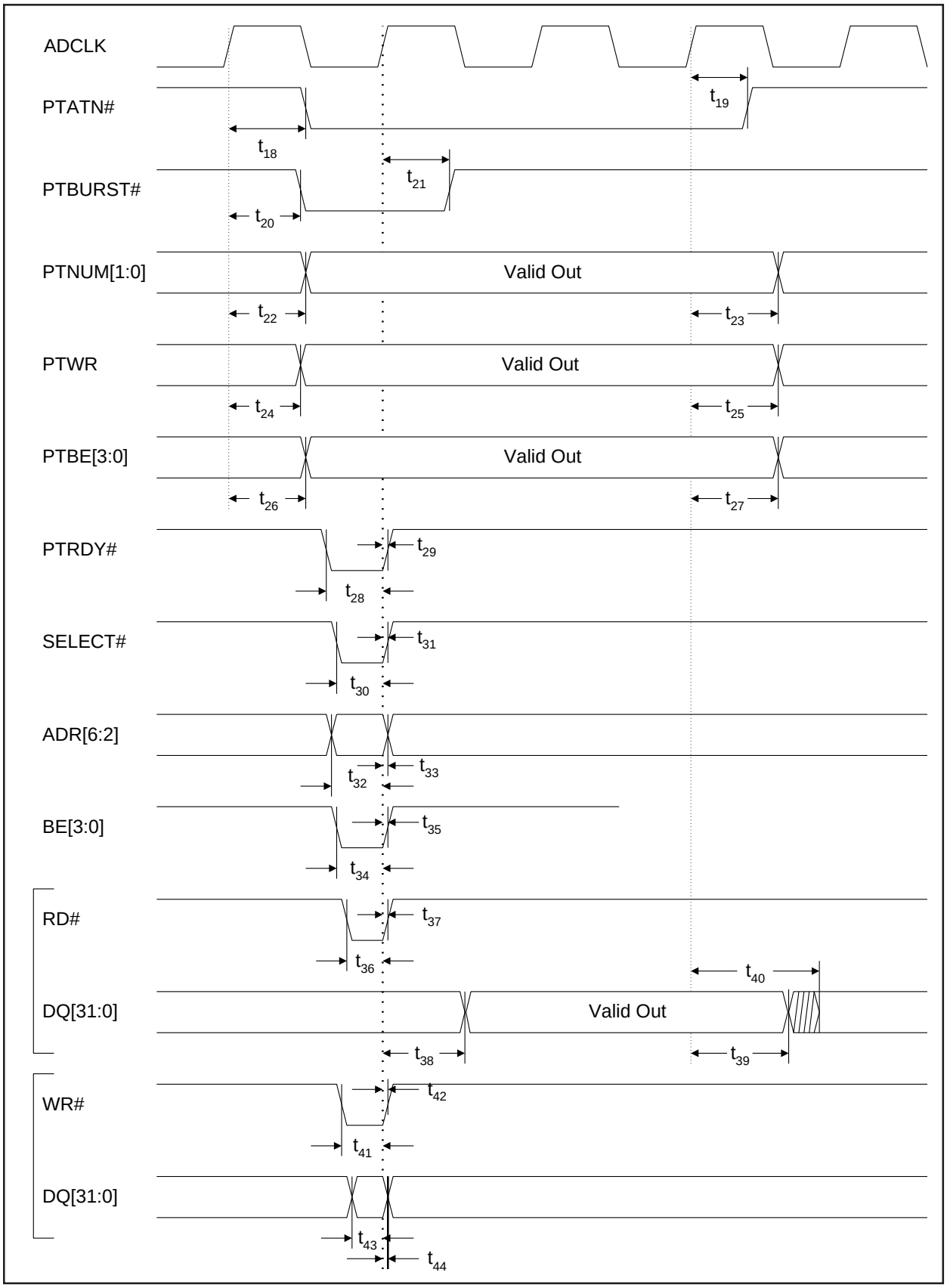


Table 7. Add-On Timings (Continued)

Functional Operation Range (V_{CC} = 5.0V +/- 5%, 0°C to 70°C, 50 pF load on outputs for MAX, 0 pF load for MIN).

Symbol	Parameter	Min	Max	Units	Notes
t_{45}	DXFER# Valid from ADCLK Rising Edge		12.5	ns	
t_{46}	DXFER# Hold from ADCLK Rising Edge	3		ns	
t_{47}	PTADR# Valid from ADCLK Rising Edge		14	ns	
t_{48}	PTADR# Hold from ADCLK Rising Edge	3		ns	
t_{49}	DQ[31:0] Address Valid from ADCLK Rising Edge		17.3	ns	1
t_{50}	DQ[31:0] Hold from ADCLK Rising Edge	3		ns	1
t_{51}	DQ[31:0] Adress Float from ADCLK Rising Edge		16.9	ns	1

Notes:

1. Refers to Pass-Thru Active mode only.

Figure 8. Active Mode Pass-Thru Write Operation

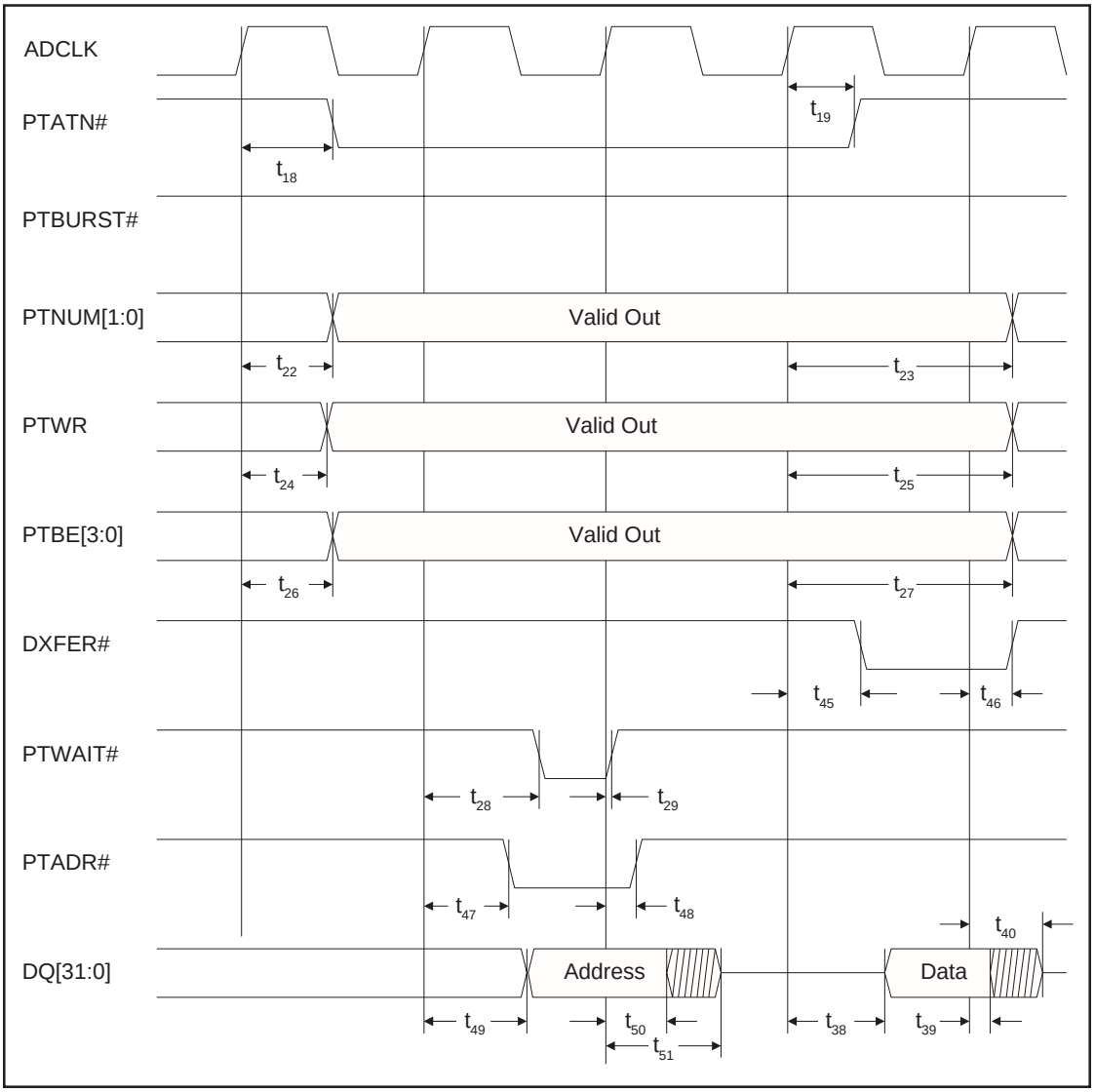


Table 8. Mailbox Timings

Functional Operation Range (V_{CC} = 5.0V +/- 5%, 0°C to 70°C, 50 pF load on outputs)

Symbol	Parameter	Min	Max	Units	Notes
t_{60}	LOAD# Setup to ADCLK Rising Edge		4	ns	
t_{61}	LOAD# Hold from ADCLK Rising Edge	1		ns	
t_{62}	MD[7:0] Setup to ADCLK Rising Edge		2.6	ns	1
t_{63}	MD[7:0] Hold from ADCLK Rising Edge	1		ns	1
t_{64}	MD[7:0] Float from LOAD# Low		14.9	ns	2
t_{65}	MD[7:0] Active from ADCLK Rising Edge		17	ns	2,3

Notes:

1. Applies only when external mailbox is in input mode (MDMODE = 1).
2. Applies only when external mailbox is in input/output mode (MDMODE = 0).
3. When the S5920 is driving MD[7:0] w/ Add-On incoming mailbox byte 3, the PCI can update this output synchronously to the PCI clock. As a result, once driving, this output is asynchronous to ADCLK.

Figure 9. Mailbox Data

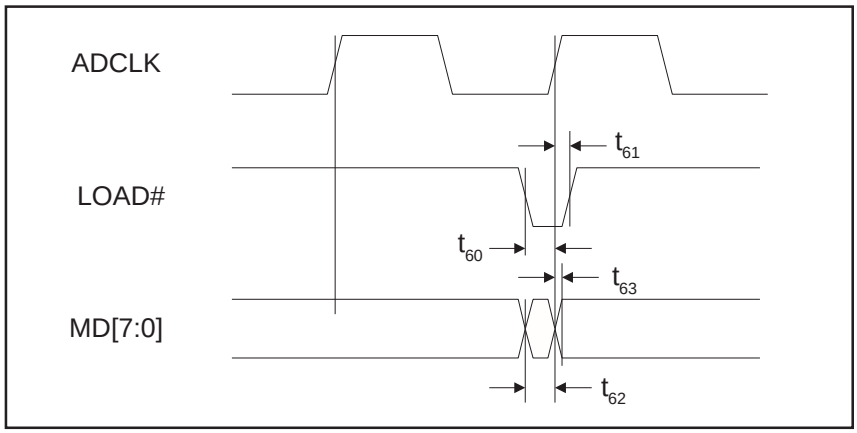


Figure 10. Mailbox Data

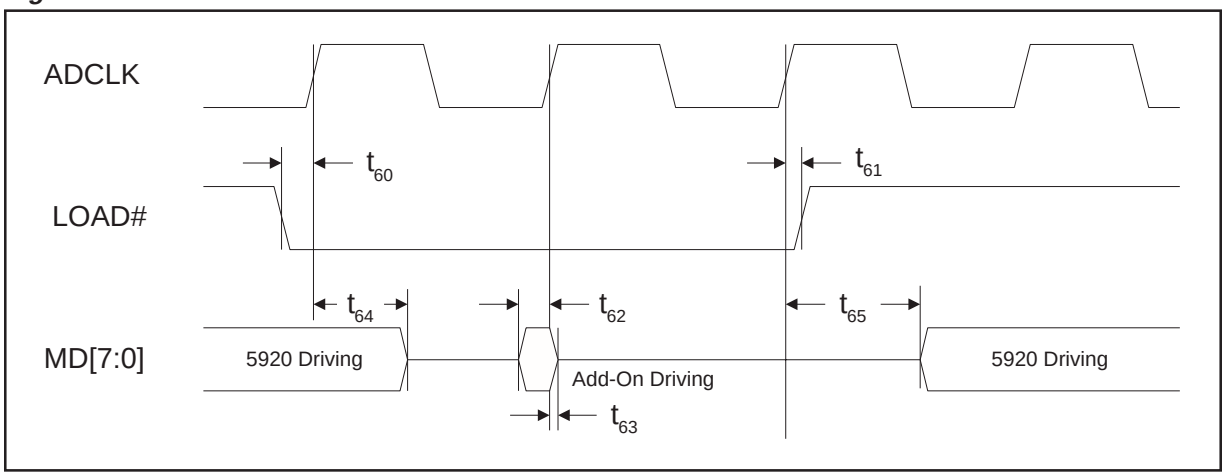


Figure 1. S5920 Pinout and Pin Assignment

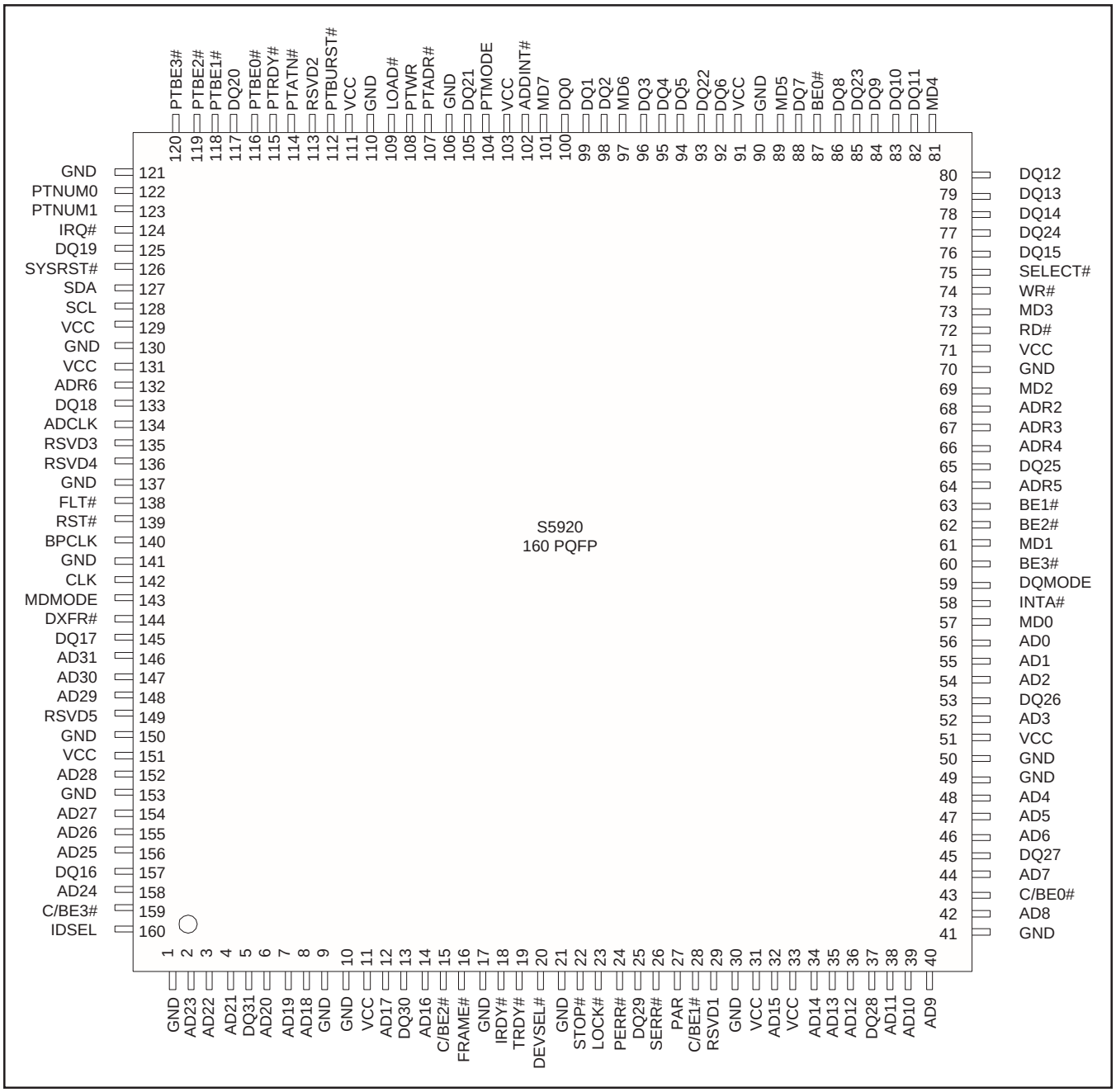
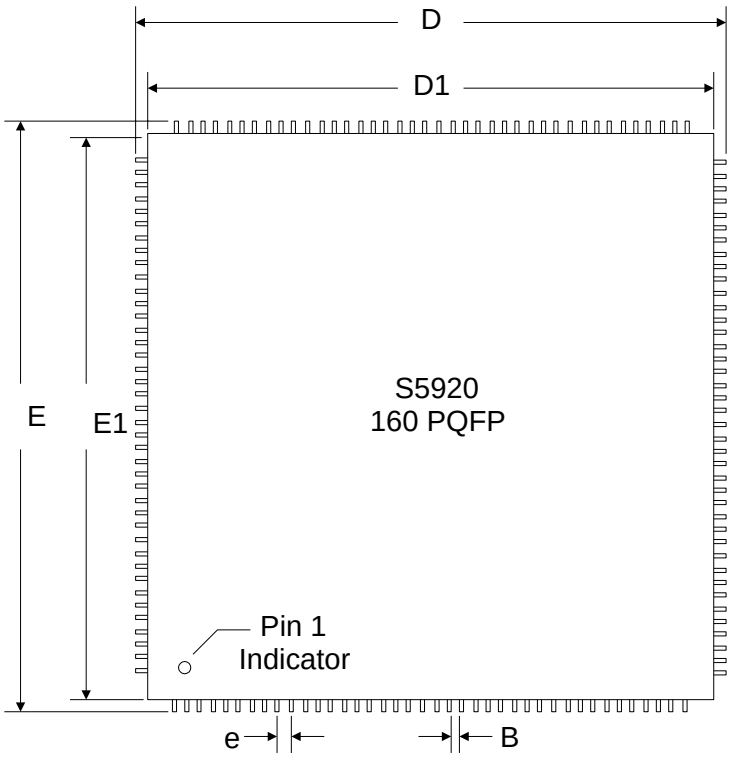
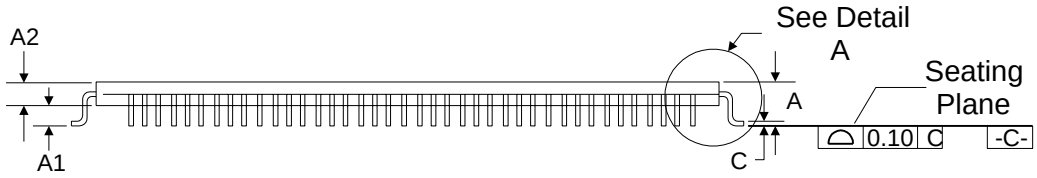
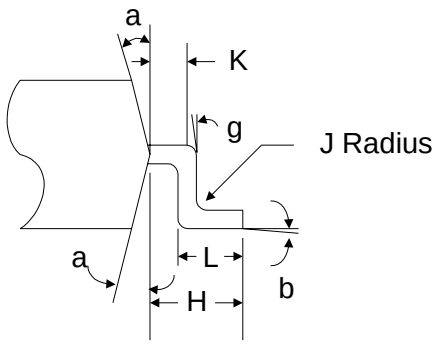


Figure 3. 160 PQFP (28 x 28 x 3.37 mm) - Plastic Quad Flat Package



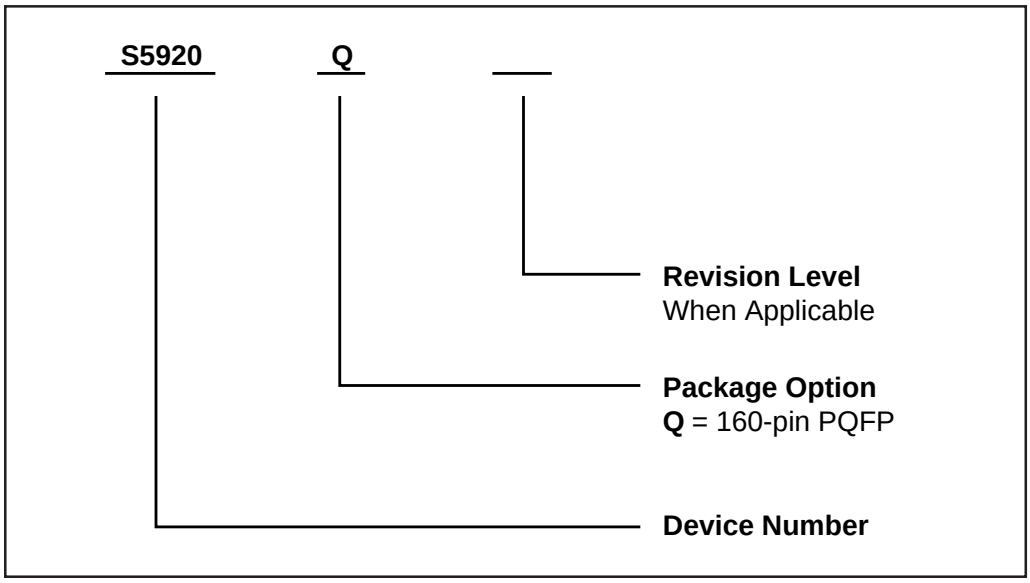
Symbol	MIN	NOM	MAX
A	-	-	4.07
A1	0.25	-	-
A2	3.17	-	-
D		31.90 BSC	
D1		28.00 BSC	
E		31.90 BSC	
E1		28.00 BSC	
L	0.65	0.80	1.03
e		0.65 BSC	
B	0.22	-	0.38
c	0.11	-	0.23
a	5	-	16
b	0	-	7
g	0	-	-
G	0.13	-	-
H		1.95 BSC	
J	0.13	-	0.30
K	0.40	-	-
2 H	-	3.9	-





Detail A

Figure 4. Ordering Information



INDEX

SECTION 3: S5933 PCI Controller

CONTENTS

INTRODUCTION TO THE PCI LOCAL BUS

The local bus concept was developed to break the PC data bottleneck. Traditional PC bus architectures are inadequate to meet the demands of today's graphics-oriented systems and large application sizes. A local bus moves peripherals off the I/O bus and places them closer to the system's processor bus, providing faster data transfer between the processor and peripherals.

The PCI Local Bus addresses the industry's need for a local bus standard that is not directly dependent on the speed and size of the processor bus, and that is both reliable and expandable. It represents the first time in the history of the PC industry that a common bus, independent of microprocessor design and manufacturer, has been adopted and used by rival PC computer architectures. PCI offers simple "plug and play" capability for the end user, and its performance is more than adequate for the most demanding applications, such as full-motion video.

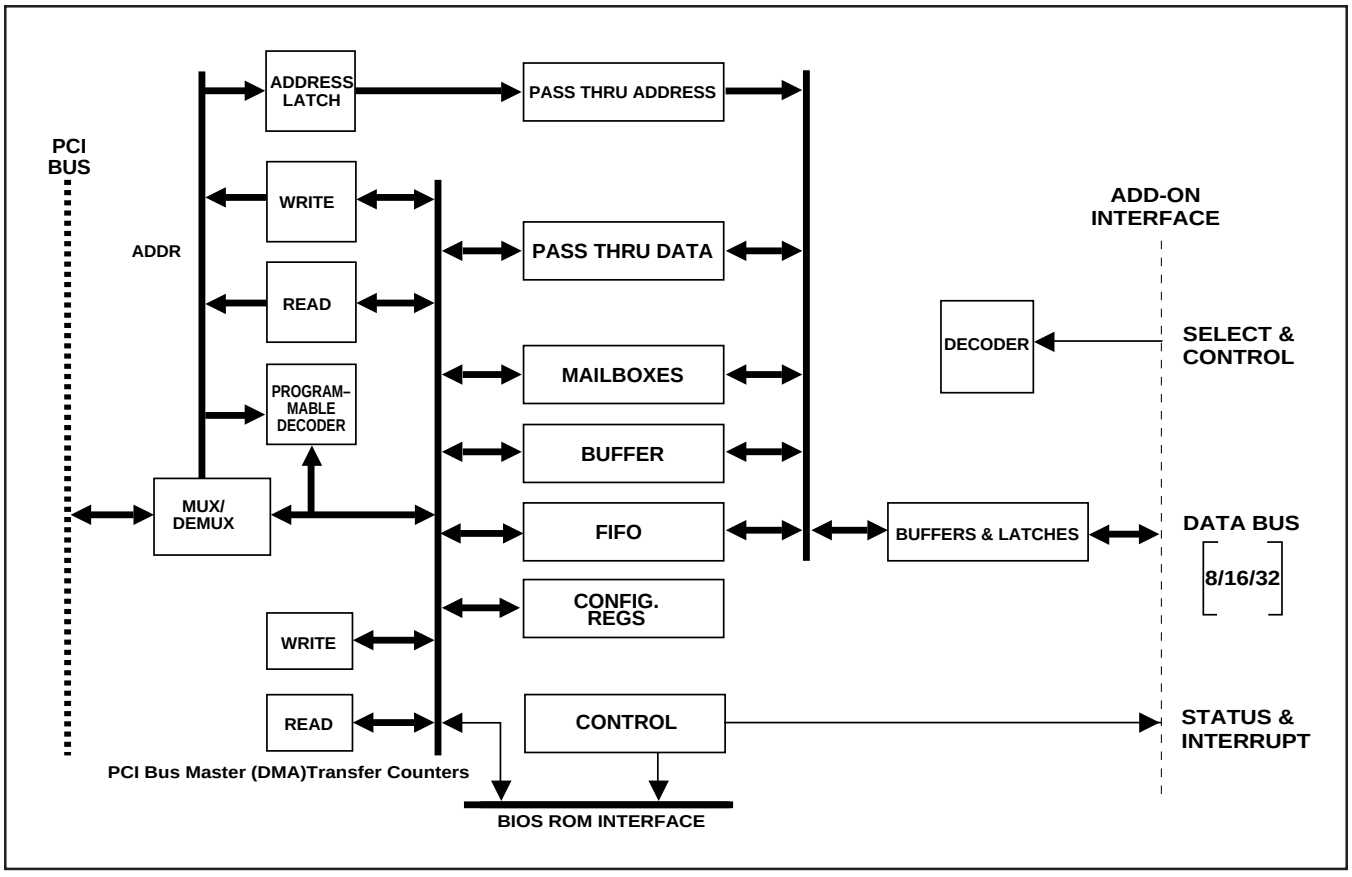
FEATURES

- PCI 2.1 Master/Slave Controller
- Generic 8/16/32-bit Add-On user bus
- Four definable memory block Pass-Thru regions
- Direct Add-On mailbox data strobe pin for PCI interrupt
- Optional boot load/external BIOS serial or byte-wide nvRAM
- Two 32 Byte FIFOs and 32 byte mailboxes
- Industry standard 160-pin PQFP

APPLICATIONS

- Digital Video
- Networking
- Multimedia
- Data Aquisition

Figure 1. S5933 Block Diagram



THE S5933 PCI CONTROLLER

AMCC's S5933 is a powerful and flexible PCI controller supporting several levels of interface sophistication. At the lowest level, it can serve simply as a PCI bus target with modest transfer requirements (since the PCI bus "plug-and-play" architecture is a benefit even when the transfer rate demand is low). For high-performance applications, the S5933 can become the bus master and can attain the peak transfer capabilities of 132 MByte/sec with a 32-bit PCI bus.

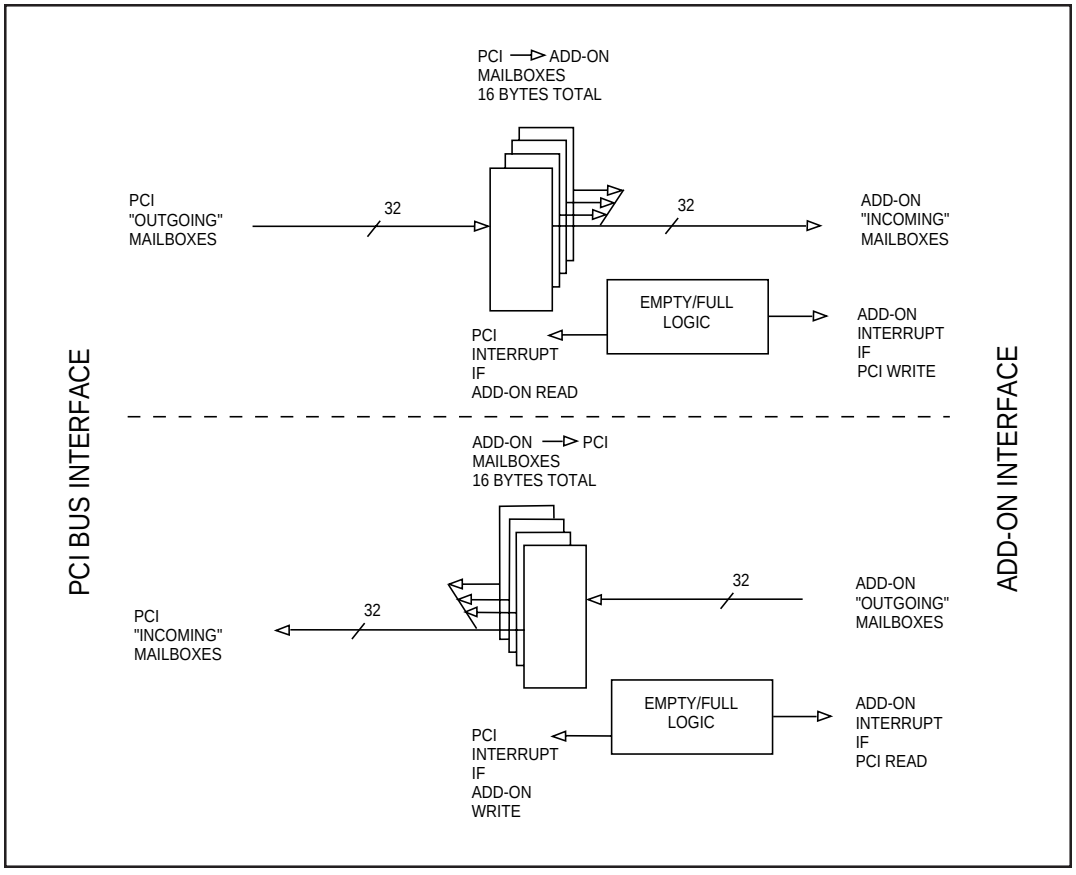
FUNCTIONAL ELEMENTS

The block diagram in Figure 1 shows the major functional elements within the S5933. The S5933 provides three physical bus interfaces: the PCI bus, the Add-On bus, and an optional external non-volatile memory. Data movement can occur between the PCI bus and the Add-On bus or the PCI bus and the non-volatile memory. Transfers between PCI and Add-On buses can take place through the mailbox registers or the FIFOs, or can make use of the Pass-Thru data path. FIFO transfers on the PCI bus interface can be performed either through software control or through hardware (using the S5933 as bus master).

MAILBOXES

The mailbox registers (shown in Figure 2) of the S5933 provide a bidirectional data path that can be used to send data or software control information between the system platform and the Add-On product. These mailboxes are intended to serve as customizable command, status, and parametric data registers. Use of the mailbox registers is defined by an application's own software; they are often used to initiate larger data transfers over either the FIFO or Pass-Thru data paths. Interrupts can be configured to occur on the PCI and Add-On bus (if desired) based on a specific mailbox event(s).

Figure 2. Mailbox Register Concept



FIFOs

Two separate FIFO data paths exist within the S5933. One FIFO handles data movement from the PCI bus to the Add-On bus (shown in Figure 3) and the other handles movement in the opposite direction (Figure 4). Both of the FIFOs are able to support PCI bus mastering; each has an address pointer and transfer count register associated with its PCI bus transaction. An important feature of the S5933 is the ability to optionally perform various endian translations when data is moved through the FIFO. This feature permits a single Add-On product to maintain a fixed endian structure within the Add-On while allowing the system platform to operate in its own native endian format.

Figure 3. PCI to Add-On FIFO Concept

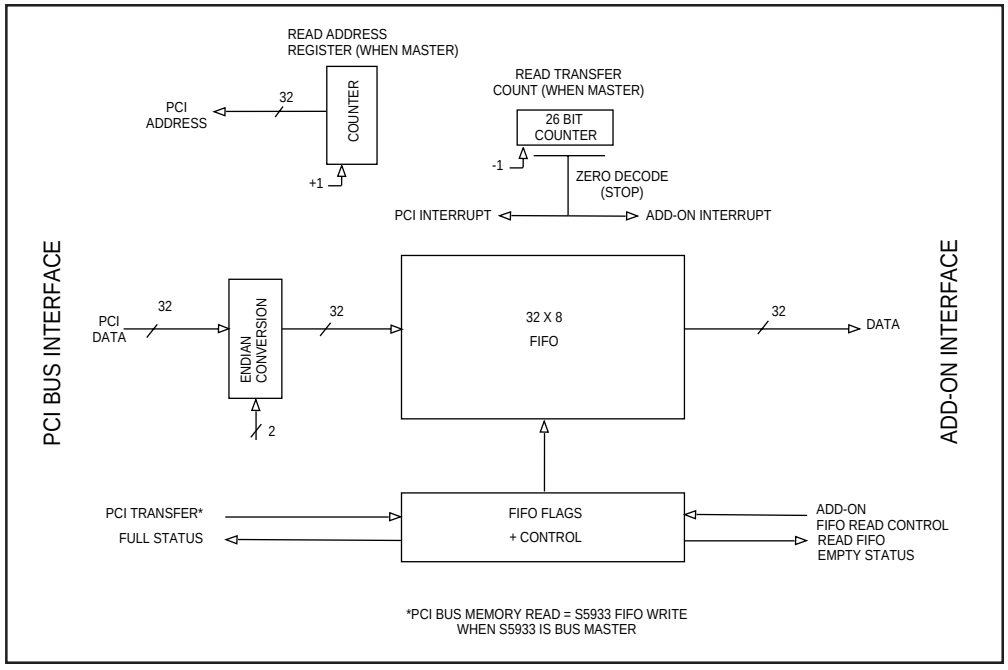
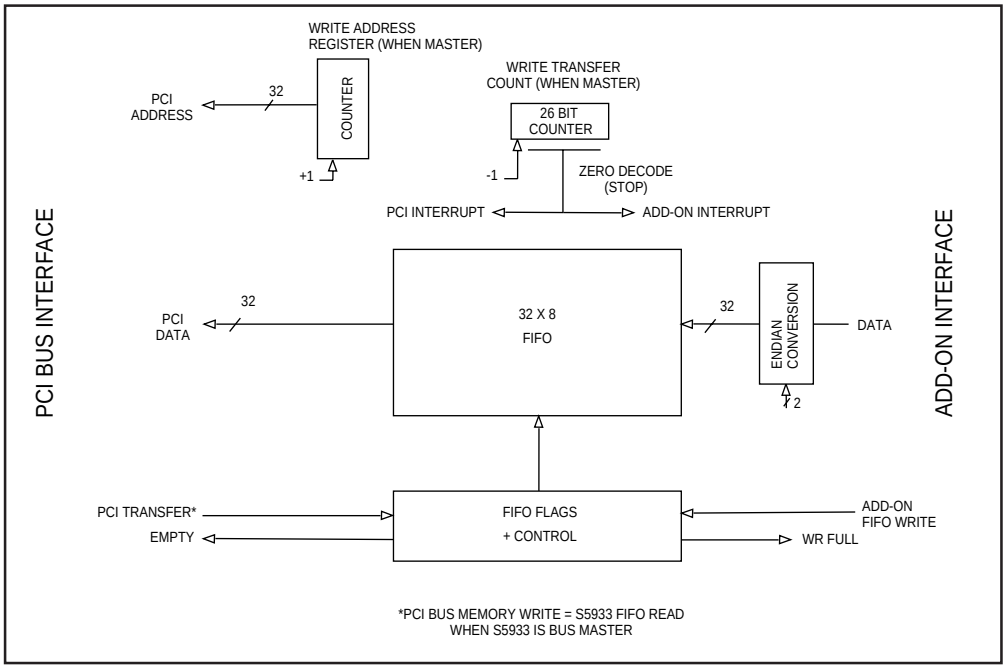


Figure 4. Add-On to PCI FIFO Concept



PASS-THRU

Figure 5 depicts the Pass-Thru concept that allows actual PCI bus transactions to be executed in real time with the Add-On interface. The Pass-Thru feature can be used to achieve high-performance burst transfers between the PCI bus and Add-On bus external peripherals or memory devices. In the Pass-Thru mode, the S5933 provides the benefits of the configuration registers and full PCI Bus Specification 2.1 compliance, while permitting customization and flexibility in the implementation of the Add-On product.

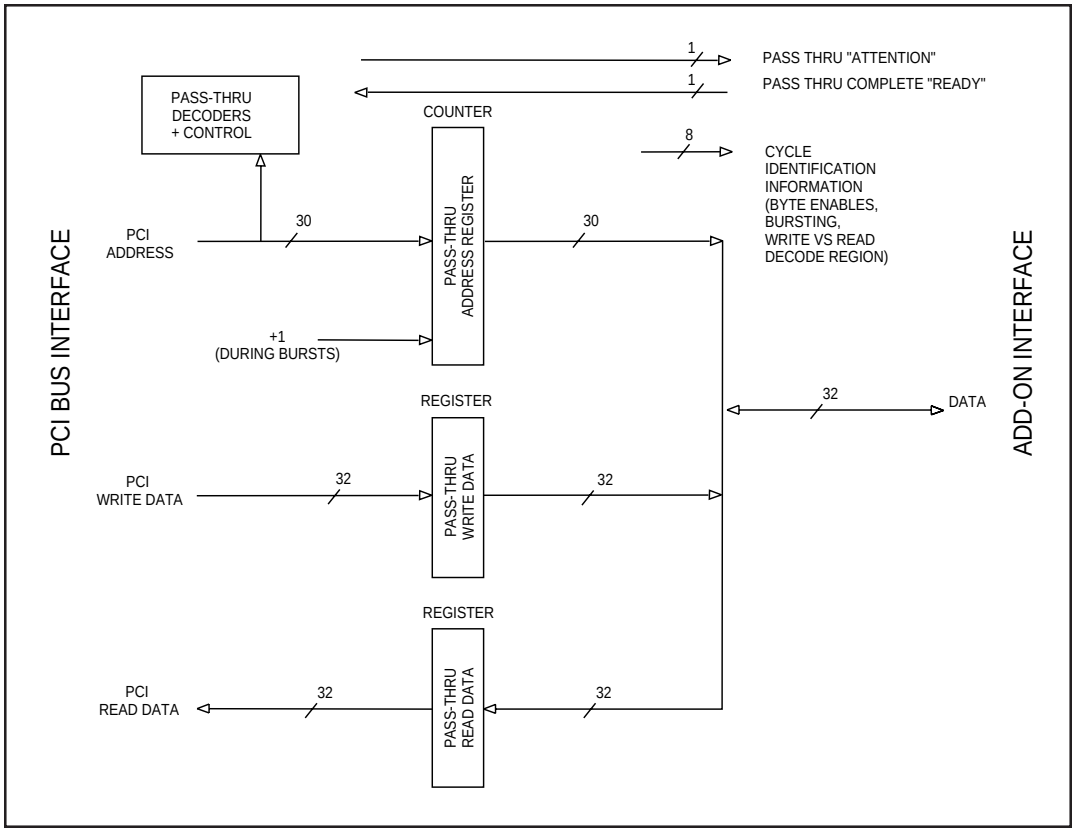
The Pass-Thru logic is comprised of one address register and two data registers (one for each direction). Control information necessary to define the current (Pass-Thru) PCI bus transaction is also provided to the Add-On interface on dedicated S5933 package pins.

PCI AGENT

The S5933 was designed to serve as an “endpoint” within a given PCI system. This means that it is the final point in a data transfer (for example, a video screen, network, sound output, or storage device) and/or the beginning point of a data transfer (for example, video source, network, audio source, or storage device). Some PCI elements, termed “bridges,” allow for data transactions to span to other bus standards and therefore are not “endpoints.”

As a PCI bus agent, the S5933 is required to support the configuration registers in order to be compliant with the PCI Specification. These configuration registers provide for a standardized method for system platform software to integrate Add-On functions in a machine. Another important aspect of the standard is that all PCI bus agents can be controlled (enabled/disabled, assigned physical address space, etc.) in a common manner, regardless of function, by the hosting system. These configuration registers defined by the PCI standard (and present within the S5933) are described in the PCI Configuration Chapter.

Figure 5. Pass-Thru Concept



ADD-ON INTERFACE

The S5933 provides a simple, general-purpose interface to the Add-On bus. The Add-On data path is a 32-bit bus for the S5933. Data transfers to/from the S5933 internal registers are accomplished through a chip select decode in conjunction with either a read or write strobe. The S5933 provides dedicated pins which allow its FIFOs to be used in the implementation of custom DMA ports or additional external FIFOs (if necessary).

The output pins on the Add-On interface include an interrupt source, a buffered clock, and a software-controllable reset. The interrupt output pin is provided to signal when a selected mailbox or self-test event occurs from the PCI interface. The buffered clock output is a possible Add-On cost reduction feature, and in addition provides synchronization for Pass-Thru data transfers. The software-controllable reset from the S5933 provides Add-On hardware with a means for proper handling of a system's "soft" reboot (e.g.CTRL-ALT-DEL).

NON-VOLATILE MEMORY INTERFACE

The non-volatile interface allows customization of the S5933 and provides for an optional BIOS ROM on the PCI bus. The non-volatile memory can be either a serial device or a byte-wide device. Serial devices may range from 128 bytes through 2048 bytes, and byte-wide devices from 128 bytes through 65,536 bytes.

As an example, the Vendor and Device ID numbers in the PCI configuration space can be initialized to reflect values contained in the external non-volatile memory. After initialization, a user's own Vendor and Device ID is then presented by the S5933 when requested by the PCI bus.

For some non-volatile devices, it is possible to write the non-volatile memory from the PCI interface. This feature lets the system designer establish a field upgrade strategy for the BIOS software or a method to support various system platforms with only one Add-On board product.

Use of a byte-wide non-volatile memory device is shown in Figure 6; Figure 7 shows a serial device with the S5933.

Figure 6. Byte-Wide Non-Volatile Memory Interface

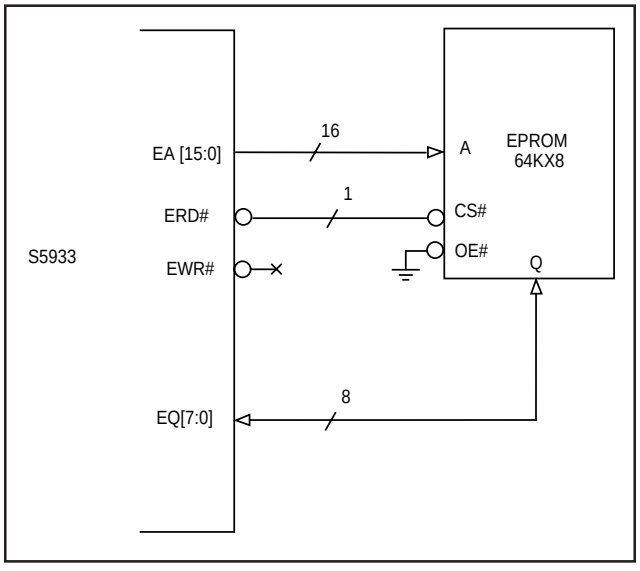


Figure 7. Serial Non-Volatile Memory Interface

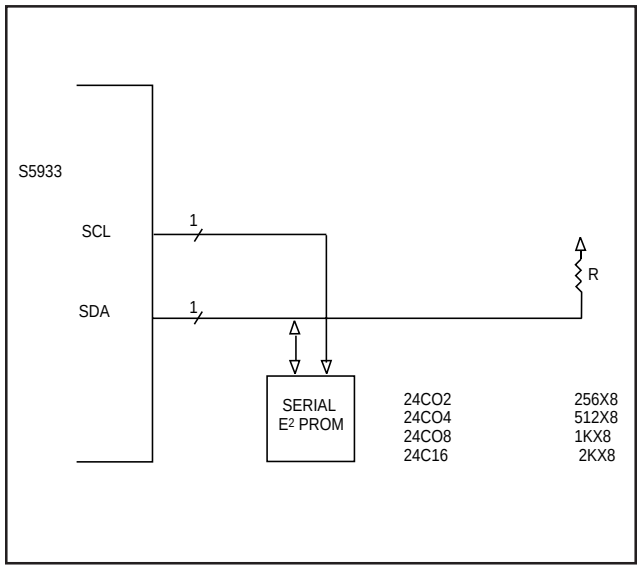
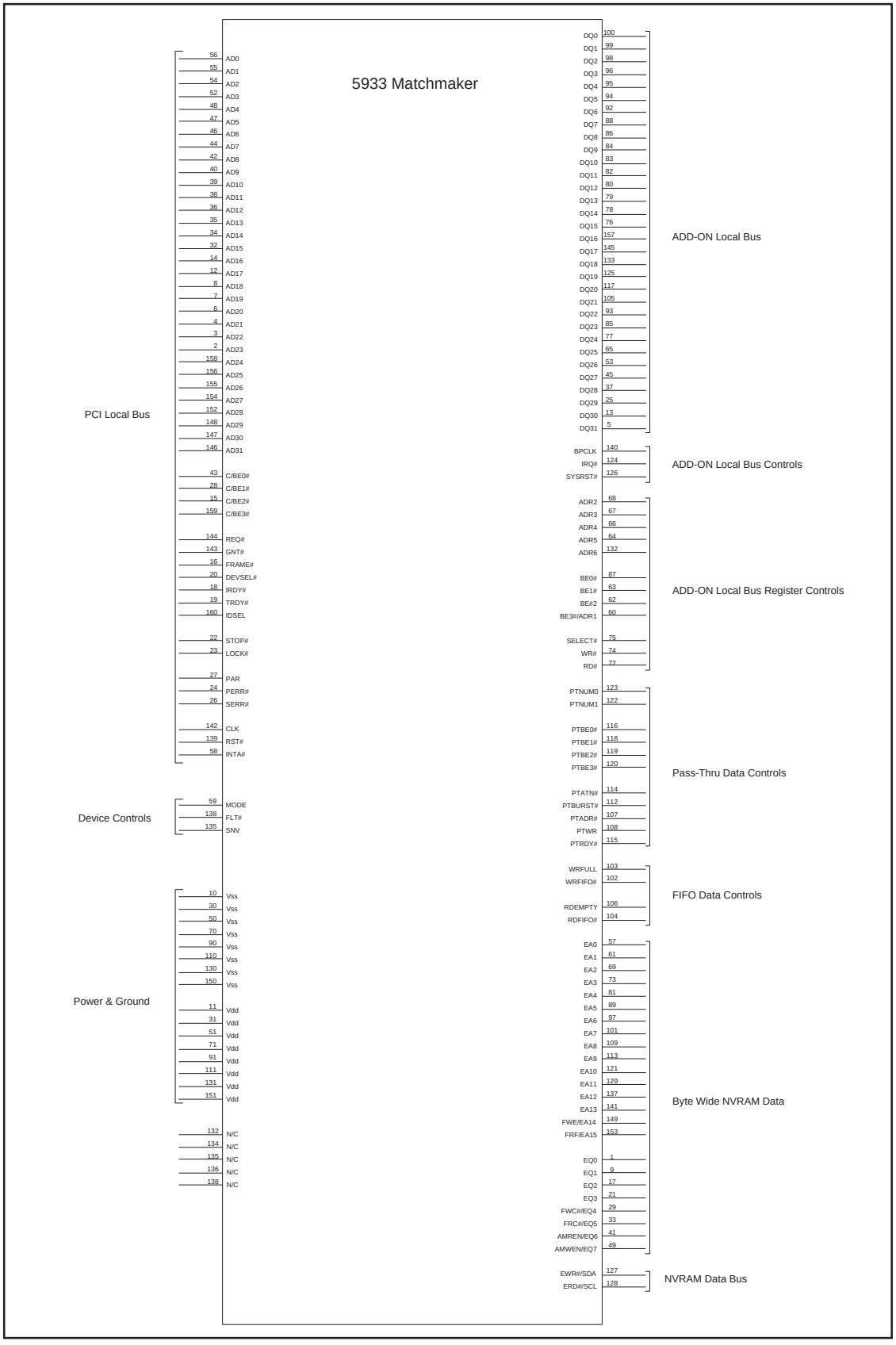


Figure 8. S5933 Pin Assignment



Signal Type Definition

The following signal type definitions [in, out, t/s, s/t/s and o/d] are taken from Revision 2.1 of the PCI local bus specification.

in Input is a standard input-only signal.

out Totem Pole Output is a standard active driver.

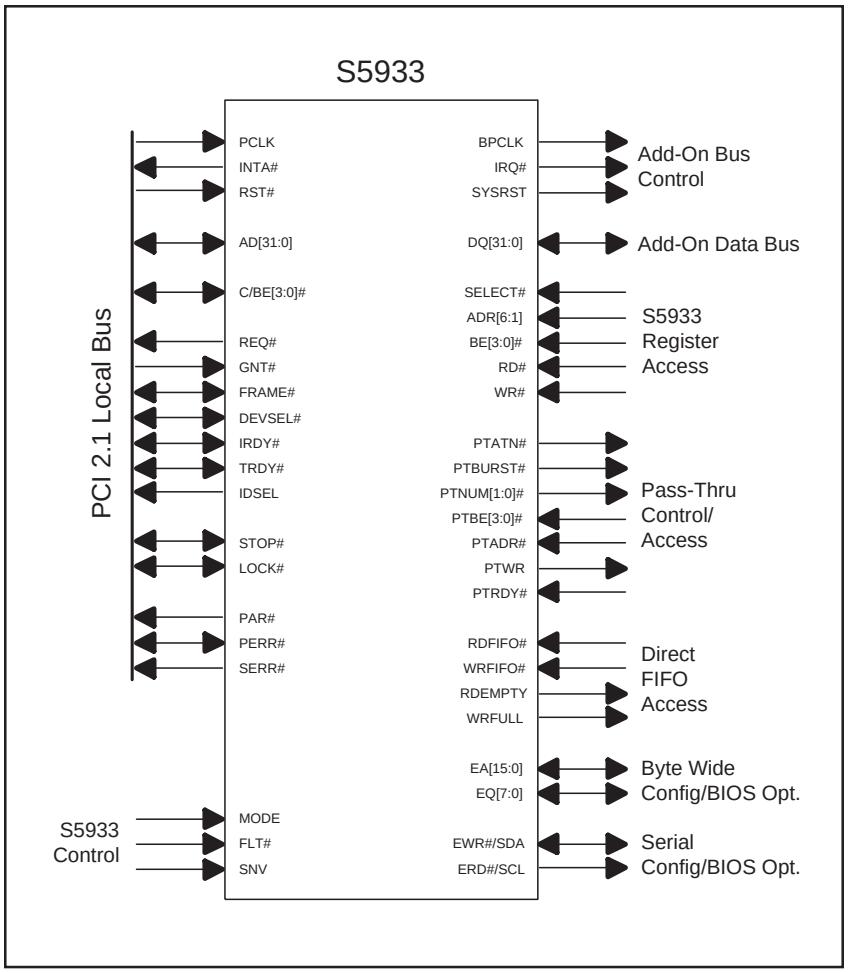
t/s Tri-State® is a bidirectional, tristate input/output pin.

s/t/s Sustained Tri-State is an active low tristate signal owned and driven by one and only one agent at a time. The agent that drives an s/t/s pin low must drive it high for at least one clock before letting it float. A new agent cannot start driving a s/t/s signal any sooner than one clock after the previous owner tri-states it. A pullup is required to sustain the inactive state until another agent drives it, and must be provided by the central source.

o/d Open Drain allows multiple devices to share as a wire-OR.

Note that a # symbol at the end of a signal name denotes that the active state occurs when the signal is at a low voltage. When no # symbol is present, the signal is active high.

Figure 1. S5933 Signal Pins



PCI BUS INTERFACE SIGNALS

Address and Data Pins — PCI Local Bus

<table>
<tr><th>Signal</th><th>Type</th><th>Description</th></tr>
<tr><td>AD[31:00]</td><td>t/s</td><td>Local Bus Address/Data lines. Address and data are multiplexed on the same pins. Each bus operation consists of an address phase followed by one or more data phases. Address phases are identified when the control signal, FRAME#, is asserted. Data transfers occur during those clock cycles in which control signals IRDY# and TRDY# are both asserted.</td></tr>
<tr><td>C/BE[3:0]#</td><td>t/s</td><td>Bus Command and Byte Enables. These are multiplexed on the same pins. During the address phase of a bus operation, these pins identify the bus command, as shown in the table below. During the data phase of a bus operation, these pins are used as Byte Enables, with C/BE[0]# enabling byte 0 (least significant byte) and C/BE[3]# enabling byte 3 (most significant byte).
<table>
<tr><th colspan="4">C/BE[3:0]#</th><th>Description
(during address phase)</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Interrupt Acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>Special Cycle</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>I/O READ</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>I/O WRITE</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>Memory Read</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>Memory Write</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>Configuration Read</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>Configuration Write</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>MEMORY READ - Multiple</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>Dual Address Cycle</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>Memory Read Line</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>Memory Write and Invalidate</td></tr>
</table></td></tr>
<tr><td>PAR</td><td>t/s</td><td>Parity. This signal is even parity across the entire AD[31:00] field along with the C/BE[3:0]# field. The parity is stable in the clock following the address phase and is sourced by the master. During the data phase for write operations, the bus master sources this signal on the clock following IRDY# active; during the data phase for read operations, this signal is sourced by the target and is valid on the clock following TRDY# active. The PAR signal therefore has the same timing as AD[31:00}, delayed by one clock.</td></tr>
</table>

System Pins — PCI Local Bus

Signal	Type	Description
CLK	in	Clock. The rising edge of this signal is the reference upon which all other signals are based, with the exception of RST# and the interrupt (IRQA#-). The maximum frequency for this signal is 33 MHz and the minimum is DC (0 Hz).
RST#	in	Reset. This signal is used to bring all other signals within this device to a known, consistent state. All PCI bus interface output signals are not driven (tri-stated), and open drain signals such as SERR# are floated.

Interface Control Pins — PCI Bus Signal

Signal	Type	Description
FRAME#	s/t/s	Frame. This signal is driven by the current bus master and identifies both the beginning and duration of a bus operation. When FRAME# is first asserted, it indicates that a bus transaction is beginning and that valid addresses and a corresponding bus command are present on the AD[31:00] and C/BE[3:0] lines. FRAME# remains asserted during the data transfer portion of a bus operation and is deasserted to signify the final data phase.
IRDY#	s/t/s	Initiator Ready. This signal is sourced by the bus master and indicates that the bus master is able to complete the current data phase of a bus transaction. For write operations, it indicates that valid data is on the AD[31:00] pins. Wait states occur until both TRDY# and IRDY# are asserted together.
TRDY#	s/t/s	Target Ready. This signal is sourced by the selected target and indicates that the target is able to complete the current data phase of a bus transaction. For read operations, it indicates that the target is providing valid data on the AD[31:00] pins. Wait states occur until both TRDY# and IRDY# are asserted together.
STOP#	s/t/s	Stop. The Stop signal is sourced by the selected target and conveys a request to the bus master to stop the current transaction.
LOCK#	in	Lock. The lock signal provides for the exclusive use of a resource. The S5933 may be locked as a target by one master at a time. The S5933 cannot lock a target when it is a master.
IDSEL	in	Initialization Device Select. This pin is used as a chip select during configuration read or write operations.
DEVSEL#	s/t/s	Device Select. This signal is sourced by an active target upon decoding that its address and bus commands are valid. For bus masters, it indicates whether any device has decoded the current bus cycle.

Arbitration Pins (Bus Masters Only) — PCI Local Bus

Signal	Type	Description
REQ#	out	Request. This signal is sourced by an agent wishing to become the bus master. It is a point-to-point signal and each master has its own REQ#.
GNT#	in	Grant. The GNT# signal is a dedicated, point-to-point signal provided to each potential bus master and signifies that access to the bus has been granted.

Error Reporting Pins — PCI Local Bus

Signal	Type	Description
PERR#	s/t/s	Parity Error. This pin is used for reporting parity errors during the data portion of a bus transaction for all cycles except a Special Cycle. It is sourced by the agent receiving data and driven active two clocks following the detection of the error. This signal is driven inactive (high) for one clock cycle prior to returning to the tri-state condition.
SERR#	o/d	System Error. This pin is used for reporting address parity errors, data parity errors on Special Cycle commands, or any error condition having a catastrophic system impact.

Interrupt Pin — PCI Local Bus

Signal	Type	Description
INTA#	o/d	Interrupt A. This pin is a level sensitive, low active interrupt to the host. The INTA# interrupt must be used for any single function device requiring an interrupt capability.

NON-VOLATILE MEMORY INTERFACE SIGNALS

This signal grouping provides for connection to external non-volatile memories. Either a serial or byte-wide device may be used.

The serial interface shares the read and write control pins used for interfacing with byte-wide memory devices. Since it is intended that only one (serial or byte wide) configuration be used in any given implementation, separate descriptions are provided for each. The S5933 provides the pins necessary to interface to a byte wide non-volatile memory. When they are connected to a properly configured serial memory, these byte wide interface pins assume alternate functions. These alternate functions include added external FIFO status flags, FIFO reset control, Add-On control for bus mastering and a hardware interface mailbox port.

Serial nv Devices

Signal	Type	Description
SCL	t/s	Serial Clock. This output is intended to drive a two-wire Serial Interface and functions as the bus's master. It is intended that this signal be directly connected to one or more inexpensive serial non-volatile RAMs or EEPROMs. This pin is shared with the byte wide interface signal, ERD#.
SDA	t/s	Serial Data/Address. This bidirectional pin is used to transfer addresses and data to or from a serial nvRAM or EEPROM. It is an open drain output and intended to be wire-ORed with all other devices on the serial bus using a 4.7K external pull-up resistor. This pin is shared with the byte wide interface signal, EWR#.
SNV	in	Serial Non-Volatile Device. This input, when high, indicates a serial boot device or no boot device is present. When this pin is low, a byte-wide boot device is present. Note: SCL and SDA are not controlled by FLT#.

Byte-Wide nv Devices

Signal	Type	Description
EA[15:00]	t/s	External nv memory address. These signals connect directly to the external BIOS (or EEPROM) or EPROM address pins EA0 through EA15. The PCI interface controller assembles 32-bit-wide accesses through multiple read cycles of the 8-bit device. The address space from 0040h through 007Fh is used to preload and initialize the PCI configuration registers. Should an external nv memory be used, the minimum size required is 128 bytes and the maximum is 64K bytes. When a serial memory is connected to the S5933, the pins EA[7:0] are reconfigured to become a hardware Add-On to PCI mailbox register with the EA8 pin as the mailbox load clock. Also, the EA15 signal pin will provide an indication that the PCI to Add-On FIFO is full (FRF), and the EA14 signal pin will indicate whether the Add-On to PCI FIFO is empty (FWE).
ERD#	out	External nv memory read control. This pin is asserted during read operations involving the external non-volatile memory. Data is transferred into the S5933 during the low to high transition of ERD#. This pin is shared with the serial external memory interface signal, SCL.
EWR#	t/s	External nv memory write control. This pin is asserted during write operations involving the external non-volatile memory. Data is presented on pins EQ[7:0] along with its address on pins EA[15:0] throughout the entire assertion of EWR#. This pin is shared with the serial external memory interface signal, SDA.
EQ[7:0]	t/s	External memory data bus. These pins are used to directly connect with the data pins of an external non-volatile memory. When a serial memory is connected to the S5933, the pins EQ4, EQ5, EQ6 and EQ7 become reconfigured to provide signal pins for bus mastering control from the Add-On interface.

ADD-ON BUS INTERFACE SIGNALS

The following sets of signals represent the interface pins available for the Add-On function. There are four groups: Register access, FIFO access, Pass-Thru mode pins, and general system pins.

Register Access Pins

Signal	Type	Description
DQ[31:00]	t/s	Datapath DQ0–DQ31. These pins represent the datapath for the Add-On peripheral's data bus. They provide the interface to the controller's FIFO and other registers. When MODE=V_{CC}, only DQ[15:00] are used. DQ[31:0] have internal pull-up resistors.
ADR[6:2]	in	Add-On Addresses. These signals are the address lines to select which of the 16 DWORD registers within the controller is desired for a given read or write cycle, as shown in the table below. ADR6 has an internal pull-down resistor.
BE3# or ADR1	in	Byte Enable 3 (32-bit mode) or ADR1 (16 bit mode). This pin is used in conjunction with the read or write strobes (RD# or WR#) and the Add-On select signal, SELECT#. As a Byte Enable, it is necessary to have this pin asserted to perform write operations to the register identified by ADR[6:2] bit locations d24 through d31; for read operations it controls the DQ[31:24] output drive. BE3# has an internal pull-up resistor.
BE[2:0]#	in	Byte Enable 2 through 0. These pins provide for individual byte control during register read or write operations. BE2# controls activity over DQ[23:DQ16], BE1# controls DQ[15:8], and BE0# controls DQ[7:0]. During read operations they control the output drive for each of their respective byte lanes; for write operations they serve as a required enable to perform the modification of each byte lane. BE[2:0]# have internal pull-up resistors.
SELECT#	in	Select for the Add-On interface. This signal must be driven low for any write or read access to the Add-On interface registers. This signal must be stable during the assertion of command signals WR# or RD#.
WR#	in	Write strobe. This pin, when asserted in conjunction with the SELECT# pin, causes the writing of one of the internal registers. The specific register and operand size are identified through address pins ADR[6:2] and the byte enables, BE[3:0]#.
RD#	in	Read strobe. This pin, when asserted in conjunction with the SELECT# pin, causes the reading of one of the internal registers. The specific register and operand size are identified through address pins ADR[6:2] and the byte enables BE[3:0]#.
MODE	in	This pin control whether the S5933 data accesses on the DQ bus are to be 32-bits wide (MODE = low) or 16-bits wide (MODE = high). When in the 16 bit mode, the signal BE3# is reassigned as the address signal ADR1. MODE has an internal pull-down resistor.

ADR[6:2]					Register Name
0	0	0	0	0	Add-On Incoming Mailbox Reg. 1
0	0	0	0	1	Add-On Incoming Mailbox Reg. 2
0	0	0	1	0	Add-On Incoming Mailbox Reg. 3
0	0	0	1	1	Add-On Incoming Mailbox Reg. 4
0	0	1	0	0	Add-On Outgoing Mailbox Reg. 1
0	0	1	0	1	Add-On Outgoing Mailbox Reg. 2
0	0	1	1	0	Add-On Outgoing Mailbox Reg. 3
0	0	1	1	1	Add-On Outgoing Mailbox Reg. 4
0	1	0	0	0	Add-On FIFO Port
0	1	0	0	1	Bus Master Write Address Register
0	1	0	1	0	Add-On Pass-Thru Address
0	1	0	1	1	Add-On Pass-Thru Data
0	1	1	0	0	Bus Master Read Address Register
0	1	1	0	1	Add-On Mailbox Empty/Full Status
0	1	1	1	0	Add-On Interrupt Control
0	1	1	1	1	Add-On General Control/Status Register
1	0	1	1	0	Bus Master Write Transfer Count
1	0	1	1	1	Bus Master Read Transfer Count

FIFO Access Pins

Signal	Type	Description
WRFIFO#	in	Write FIFO. This signal provides a method to directly write the FIFO without having to generate the SELECT# signal or the ADR[6:2] value of [01000b] to access the FIFO. Access width is either 32 bits or 16 bits depending on the data bus size available. This signal is intended for implementing PCI DMA transfers with the Add-On system. This pin has an internal pull-up resistor.
RDFIFO#	in	Read FIFO. This signal provides a method to directly read the FIFO without having to generate the SELECT# signal or the ADR[6:2] value of [01000b] to access the FIFO. Access width is either 32 bits or 16 bits, depending on the data bus size defined by the MODE pin. This signal is intended for implementing PCI DMA transfers with the Add-On system. This pin has an internal pull-up resistor.
WRFULL	out	Write FIFO full. This pin indicates whether the Add-On-to-PCI bus FIFO is able to accept more data. This pin is intended to be used to implement DMA hardware on the Add-On system bus. A logic low output from this pin can be used to represent a DMA write (Add-On to-PCI FIFO) request.
RDEMPTY	out	Read FIFO Empty. This pin indicates whether the read FIFO (PCI-to-Add-On FIFO) contains data. This pin is intended to be used by the Add-On system to control DMA transfers from the PCI bus to the Add-On system bus. A logic low from this pin can be used to represent a DMA (PCI-to-Add-On FIFO) request.

Pass-Thru Interface Pins

Signal	Type	Description
PTATN#	out	Pass-Thru Attention. This signal identifies that an active PCI bus cycle has been decoded and data must be read from or written to the Pass-Thru Data Register.
PTBURST#	out	Pass-Thru Burst. This signal identifies PCI bus operations involving the current Pass-Thru cycle as requesting burst access.
PTRDY#	in	Pass-Thru Ready. This input indicates when Add-On logic has completed a Pass-Thru cycle and another may be initiated.
PTNUM[1:0]	out	Pass-Thru Number. These signals identify which of the four base address registers decoded a Pass-Thru bus activity. These bits are only meaningful when signal PTATN# is active. A value of 00 corresponds to Base Address Register 1, a value of 01 for Base Address Register 2, and so on.
PTBE[3:0]#	out	Pass-Thru Byte Enables. These signals indicate which bytes are requested for a given Pass-Thru operation. They are valid during the presence of signal PTATN# active.
PTADR#	in	Pass-Thru Address. This signal causes the actual Pass-Thru requested address to be presented as outputs on the DQ pins DQ[31:0] for Add-Ons with 32-bit buses, or the low-order 16 bits for Add-Ons with 16-bit buses. It is necessary that all other bus control signals be in their inactive state during the assertion of PTADR#. The purpose of this signal is to provide the direct addressing of external Add-On peripherals through use of the PTNUM[1:0] and the low-order address bits presented on the DQ bus with this pin active.
PTWR	out	Pass-Thru Write. This signal identifies whether a Pass-Thru operation is a read or write cycle. This signal is valid only when PTATN# is active.

System Pins

Signal	Type	Description
SYSRST#	out	System Reset. This low active output is a buffered form of the PCI bus reset, RST#. It is not synchronized to any clock within the PCI interface controller. Additionally, this signal can be invoked through software from the PCI host interface.
BPCLK	out	Buffered PCI Clock. This output is a buffered form of the PCI bus clock and, as such, has all of the behavioral characteristics of the PCI clock (i.e., DC-to-33 MHz capability).
IRQ#	out	Interrupt. This pin is used to signal the Add-On system that a significant event has occurred as a result of activity within the PCI controller.
FLT#	in	Float. When asserted, all S5933 outputs are floated. This pin has an internal pull-up resistor.

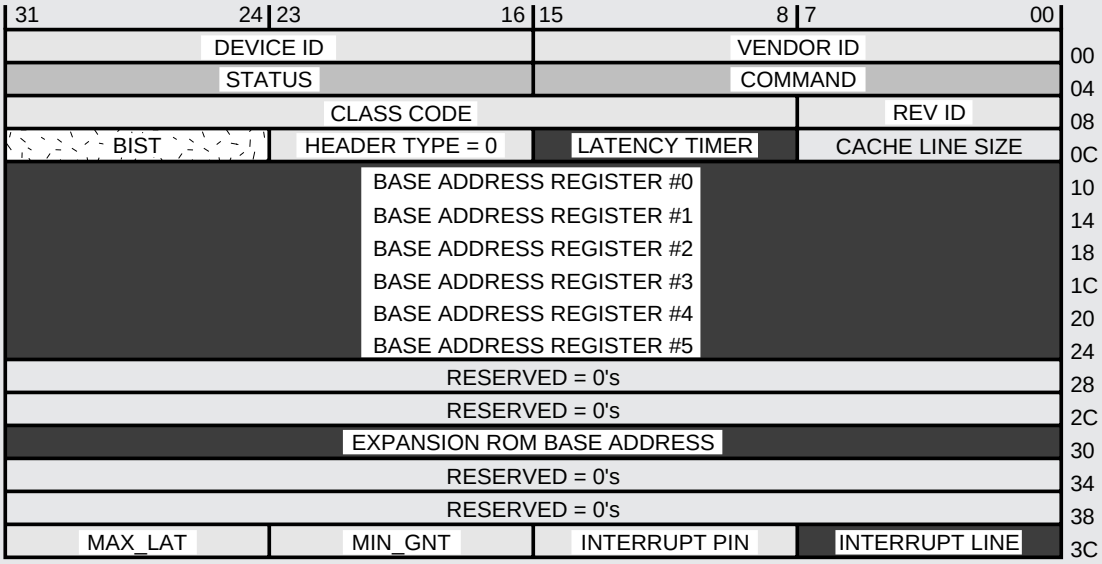
PCI CONFIGURATION REGISTERS

Each PCI bus device contains a unique 256-byte region called its configuration header space. Portions of this configuration header are mandatory in order for a PCI agent to be in full compliance with the PCI specification. This section describes each of the configuration space fields—its address, default values, initialization options, and bit definitions—and also provides an explanation of its intended usage.

Table 1. Configuration Registers

Configuration Address Offset	Abbreviation	Register Name
00h–01h	VID	Vendor Identification
02h–03h	DID	Device Identification
04h–05h	PCICMD	PCI Command Register
06h–07h	PCISTS	PCI Status Register
08h	RID	Revision Identification Register
09h–0Bh	CLCD	Class Code Register
0Ch	CALN	Cache Line Size Register
0Dh	LAT	Master Latency Timer
0Eh	HDR	Header Type
0Fh	BIST	Built-in Self-test
10h–27h	BADR0-BADR5	Base Address Registers (0-5)
28h–2Fh	—	Reserved
30h	EXROM	Expansion ROM Base Address
34h–3Bh	—	Reserved
3Ch	INTLN	Interrupt Line
3Dh	INTPIN	Interrupt Pin
3Eh	MINGNT	Minimum Grant
3Fh	MAXLAT	Maximum Latency
40h–FFh	—	Not used

PCI Configuration Space Header



LEGEND

- EPROM IS DATA SOURCE (READ ONLY)
- CONTROL FUNCTION
- EPROM INITIALIZED RAM (CAN BE ALTERED FROM PCI PORT)
- EPROM INITIALIZED RAM (CAN BE ALTERED FROM ADD-ON PORT)
- HARD-WIRED TO ZEROES

Note: Some registers are a combination of the above. See individual sections for full description.

VENDOR IDENTIFICATION REGISTER (VID)

Register Name: Vendor Identification
Address Offset: 00h-01h
Power-up value: 10E8h (AMCC, Applied Micro Circuits Corp.)
Boot-load: External nvRAM offset 040h-41h
Attribute: Read Only (RO)
Size: 16 bits

The VID register contains the vendor identification number. This number is assigned by the PCI Special Interest Group and is intended to uniquely identify any PCI device. Write operations from the PCI interface have no effect on this register. After reset is removed, this field can be boot-loaded from the external non-volatile device (if present and valid) so that other legitimate PCI SIG members can substitute their vendor identification number for this field.

Figure 1. Vendor Identification Register

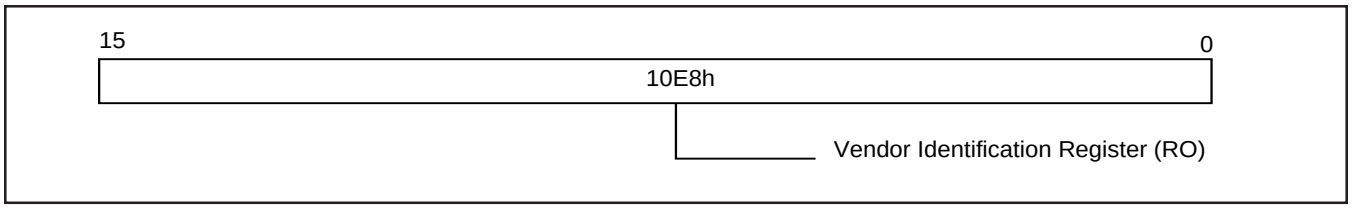


Table 2. Vendor Identification Register

Bit	Description
15:0	Vendor Identification Number: This is a 16 bit-value assigned to AMCC.

DEVICE IDENTIFICATION REGISTER (DID)

Register Name: Device Identification
Address Offset: 02h-03h
Power-up value: 4750h (ASCII hex for 'GP', General Purpose)
Boot-load: External nvRAM offset 042h-43h
Attribute: Read Only
Size: 16 bits

The DID register contains the vendor-assigned device identification number. This number is generated by AMCC in compliance with the conditions of the PCI specification. Write operations from the PCI interface have no effect on this register. After reset is removed, this field can be boot-loaded from the external non-volatile device (if present and valid) so that other legitimate PCI SIG members can substitute their own device identification number for this field.

Figure 2. Device Identification Register

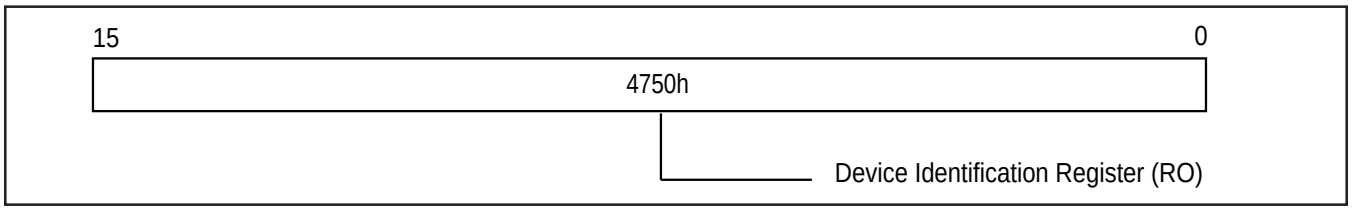


Table 3. Device Identification Register

Bit	Description
15:0	Device Identification Number: This is a 16-bit value initially assigned by AMCC for applications using the AMCC Vendor ID.

PCI COMMAND REGISTER (PCICMD)

Register Name: PCI Command
Address Offset: 04h-05h
Power-up value: 0000h
Boot-load: not used
Attribute: Read/Write (R/W on 6 bits, Read Only for all others)
Size: 16 bits

This 16-bit register contains the PCI Command. The function of this register is defined by the PCI specification and its implementation is required of all PCI devices. Only six of the ten fields are used by this device; those which are not used are hardwired to 0. The definitions for all fields are provided here for completeness.

Figure 3. PCI Command Register

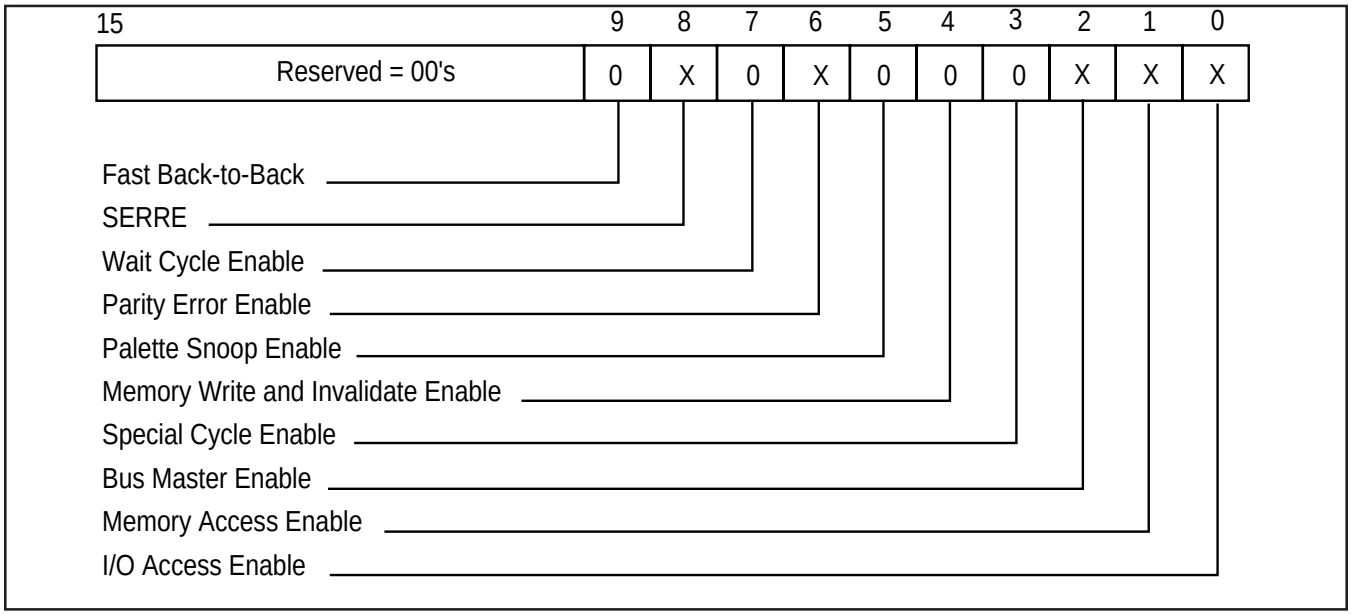


Table 4. PCI Command Register

Bit	Description
15:10	Reserved. Equals all 0's.
9	Fast Back-to-Back Enable. The S5933 does not support this function. This bit must be set to zero. This bit is cleared to a 0 upon RESET#.
8	System Error Enable. When this bit is set to 1, it permits the S5933 controller to drive the open drain output pin, SERR#. This bit is cleared to 0 upon RESET#. The SERR# pin driven active normally signifies a parity error on the address/control bus.
7	Wait Cycle Enable. This bit controls whether this device does address/data stepping. Since the S5933 controller never uses stepping, it is hardwired to 0.
6	Parity Error Enable. This bit, when set to a one, allows this controller to check for parity errors. When a parity error is detected, the PCI bus signal PERR# is asserted. This bit is cleared (parity testing disabled) upon the assertion of RESET#.
5	Palette Snoop Enable. This bit is not supported by the S5933 controller and is hardwired to 0. This feature is used solely for PCI-based VGA devices.
4	Memory Write and Invalidate Enable. This bit allows certain Bus Master devices to use the Memory Write and Invalidate PCI bus command when set to 1. When set to 0, masters must use the Memory Write command instead. The S5933 controller does not support this command when operated as a master and therefore it is hardwired to 0.
3	Special Cycle Enable. Devices which are capable of monitoring special cycles can do so when this bit is set to 1. The S5933 controller does not monitor (or generate) special cycles and this bit is hardwired to 0.
2	Bus Master Enable. This bit, when set to a one, allows the S5933 controller to function as a bus master. This bit is initialized to 0 upon the assertion of signal pin RESET#.
1	Memory Space Enable. This bit allows the S5933 controller to decode and respond as a target for memory regions that may be defined in one of the five base address registers. This bit is initialized to 0 upon the assertion of signal pin RESET#.
0	I/O Space Enable. This bit allows the S5933 controller to decode and respond as a target to I/O cycles which are to regions defined by any one of the five base address registers. This bit is initialized to 0 upon the assertion of signal pin RESET#.

PCI STATUS REGISTER (PCISTS)

Register Name: PCI Status

Address Offset: 06h-07h

Power-up value: 0080h

Boot-load: not used

Attribute: Read Only (RO), Read/Write Clear (R/WC)

Size: 16 bits

This 16-bit register contains the PCI status information. The function of this register is defined by the PCI specification and its implementation is required of all PCI devices. Only some of the bits are used by this device; those which are not used are hardwired to 0. Most status bits within this register are designated as “write clear,” meaning that in order to clear a given bit, the bit must be written as a 1. All bits written with a 0 are left unchanged. These bits are identified in Figure 4 as (R/WC). Those which are Read Only are shown as (RO) in Figure 4.

Figure 4. PCI Status Register

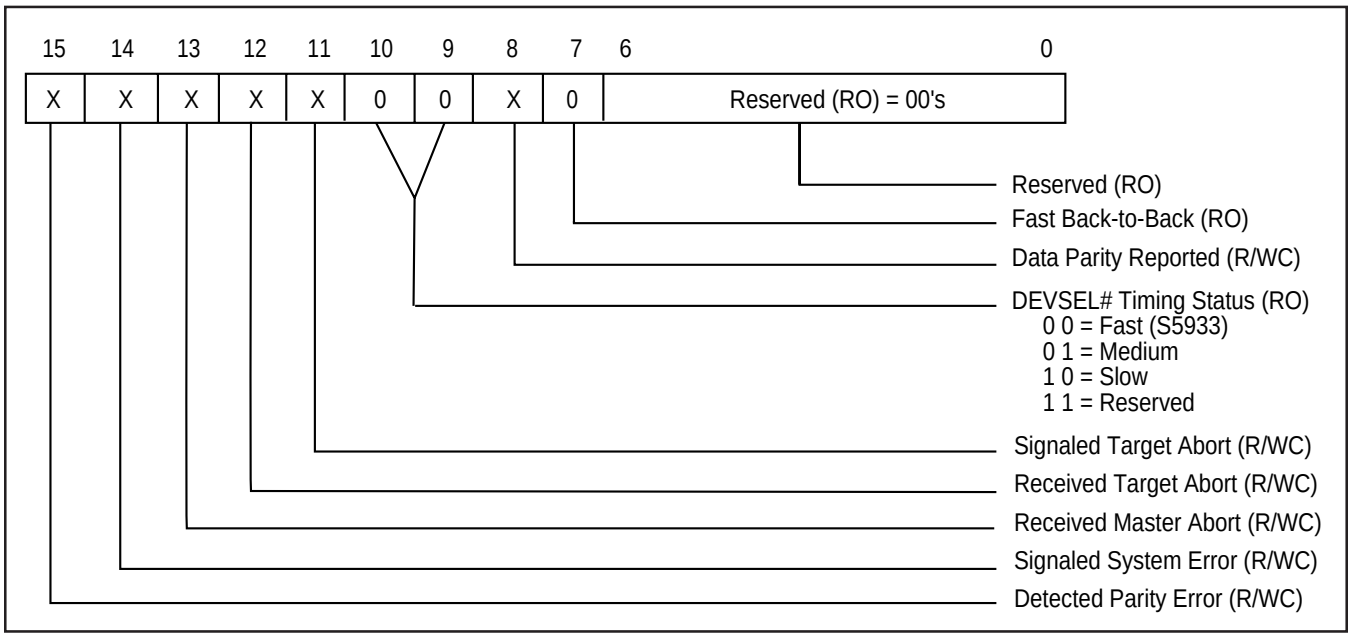


Table 5. PCI Status Register

Bit	Description
15	Detected Parity Error. This bit is set whenever a parity error is detected. It functions independently from the state of Command Register Bit 6. This bit may be cleared by writing a 1 to this location.
14	Signaled System Error. This bit is set whenever the device asserts the signal SERR#. This bit can be reset by writing a 1 to this location.
13	Received Master Abort. This bit is set whenever a bus master abort occurs. This bit can be reset by writing a 1 to this location.
12	Received Target Abort. This bit is set whenever this device has one of its own initiated cycles terminated by the currently addressed target. This bit can be reset by writing a 1 to this location.
11	Signaled Target Abort. This bit is set whenever this device aborts a cycle when addressed as a target. This bit can be reset by writing a 1 to this location.
10:9	Device Select Timing. These bits are read-only and define the signal behavior of DEVSEL# from this device when accessed as a target.
8	Data Parity Reported. This bit is set upon the detection of a data parity error for a transfer involving the S5933 device as the master. The Parity Error Enable bit (D6 of the Command Register) must be set in order for this bit to be set. Once set, it can only be cleared by either writing a 1 to this location or by the assertion of the signal RESET#.
7	Fast Back-to-back Capable. When equal to 1, this indicates that the device can accept fast back-to-back cycles as a target.
6:0	Reserved. Equal all 0's.

REVISION IDENTIFICATION REGISTER (RID)

Register Name: Revision Identification
Address Offset: 08h
Power-up value: 00h
Boot-load: External nvRAM/EPROM offset 048h
Attribute: Read Only
Size: 8 bits

The RID register contains the revision identification number. This field is initially cleared. Write operations from the PCI interface have no effect on this register. After reset is removed, this field can be boot-loaded from the external non-volatile device (if present and valid) so that another value may be used.

Figure 5. Revision Identification Register

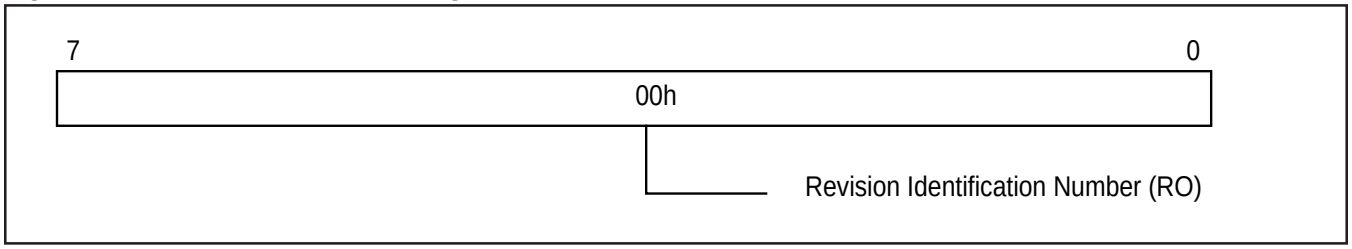


Table 6. Revision Identification Register

Bit	Description
7:0	Revision Identification Number. Initialized to zeros, this register may be loaded to the value in non-volatile memory at offset 048h.

CLASS CODE REGISTER (CLCD)

Register Name: Class Code
Address Offset: 09h-0Bh
Power-up value: FF0000h
Boot-load: External nvRAM offset 049h-4Bh
Attribute: Read Only
Size: 24 bits

This 24-bit, read-only register is divided into three one-byte fields: the base class resides at location 0Bh, the sub-class at 0Ah, and the programming interface at 09h. The default setting for the base class is all ones (FFh), which indicates that the device does not fit into the thirteen base classes defined in the PCI Local Bus Specification. It is possible, however, through use of the external non-volatile memory, to implement one of the defined class codes described in Table 7 below.

For devices that fall within the seven defined class codes, sub-classes are also assigned. Tables 8 through 20 describe each of the sub-class codes for base codes 00h through 0Ch, respectively.

Figure 6. Class Code Register

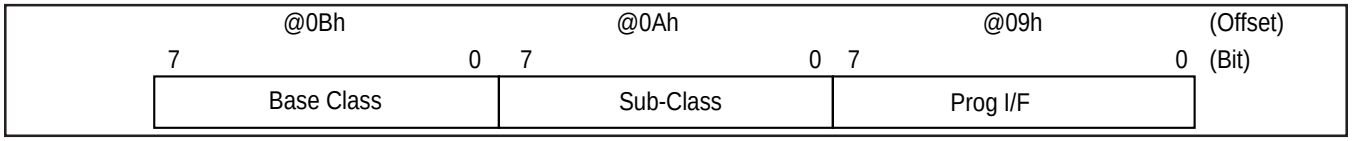


Table 7. Defined Base Class Codes

Base-Class	Description
00h	Early, pre-2.0 PCI specification devices
01h	Mass storage controller
02h	Network controller
03h	Display controller
04h	Multimedia device
05h	Memory controller
06h	Bridge device
07h	Simple communication controller
08h	Base system peripherals
09h	Input devices
0Ah	Docking stations
0Bh	Processors
0Ch	Serial bus controllers
0D-FEh	Reserved
FFh	Device does not fit defined class codes (default)

Table 8. Base Class Code 00h: Early, Pre-2.0 Specification Devices

Sub-Class	Prog I/F	Description
00h	00h	All devices other than VGA
01h	00h	VGA-compatible device

Table 9. Base Class Code 01h: Mass Storage Controllers

Sub-Class	Prog I/F	Description
00h	00h	SCSI controller
01h	xxh	IDE controller
02h	00h	Floppy disk controller
03h	00h	IPI controller
04h	00h	RAID controller
80h	00h	Other mass storage controller

Table 10. Base Class Code 02h: Network Controllers

Sub-Class	Prog I/F	Description
00h	00h	Ethernet controller
01h	00h	Token ring controller
02h	00h	FDDI controller
03h	00h	ATM controller
80h	00h	Other network controller

Table 11. Base Class Code 03h: Display Controllers

Sub-Class	Prog I/F	Description
00h	00h	VGA-compatible controller
00h	01h	8514 compatible controller
01h	00h	XGA controller
80h	00h	Other display controller

Table 12. Base Class Code 04h: Multimedia Devices

Sub-Class	Prog I/F	Description
00h	00h	Video device
01h	00h	Audio device
80h	00h	Other multimedia device

Table 13. Base Class Code 05h: Memory Controllers

Sub-Class	Prog I/F	Description
00h	00h	RAM memory controller
01h	00h	Flash memory controller
80h	00h	Other memory controller

Table 14. Base Class Code 06h: Bridge Devices

Sub-Class	Prog I/F	Description
00h	00h	Host/PCI bridge
01h	00h	PCI/ISA bridge
02h	00h	PCI/EISA bridge
03h	00h	PCI/Micro Channel bridge
04h	00h	PCI/PCI bridge
05h	00h	PCI/PCMCIA bridge
06h	00h	NuBus bridge
07h	00h	CardBus bridge
80h	00h	Other bridge type

Table 15. Base Class Code 07h: Simple Communications Controllers

Sub-Class	Prog I/F	Description
00h	00h	Generic XT compatible serial controller
	01h	16450 compatible serial controller
	02h	16550 compatible serial controller
01h	00h	Parallel port
	01h	Bidirectional parallel port
	02h	ECP 1.X compliant parallel port
80h	00h	Other communications device

Table 16. Base Class Code 08h: Base System Peripherals

Sub-Class	Prog I/F	Description
00h	00h	Generic 8259 PIC
	01h	ISA PIC
	02h	EISA PIC
01h	00h	Generic 8237 DMA controller
	01h	ISA DMA controller
	02h	EISA DMA controller
02h	00h	Generic 8254 system timer
	01h	ISA system timer
	02h	EISA system timers (2 timers)
03h	00h	Generic RTC controller
	01h	ISA RTC controller
80h	00h	Other system peripheral

Table 17. Base Class Code 09h: Input Devices

Sub-Class	Prog I/F	Description
00h	00h	Keyboard controller
01h	00h	Digitizer (Pen)
02h	00h	Mouse controller
80h	00h	Other input controller

Table 18. Base Class Code 0Ah: Docking Stations

Sub-Class	Prog I/F	Description
00h	00h	Generic docking station
80h	00h	Other type of docking station

Table 19. Base Class Code 0Bh: Processors

Sub-Class	Prog I/F	Description
00h	00h	Intel386™
01h	00h	Intel486™
02h	00h	Pentium™
10h	00h	Alpha™
40h	00h	Co-processor

Table 20. Base Class Code 0Ch: Serial Bus Controllers

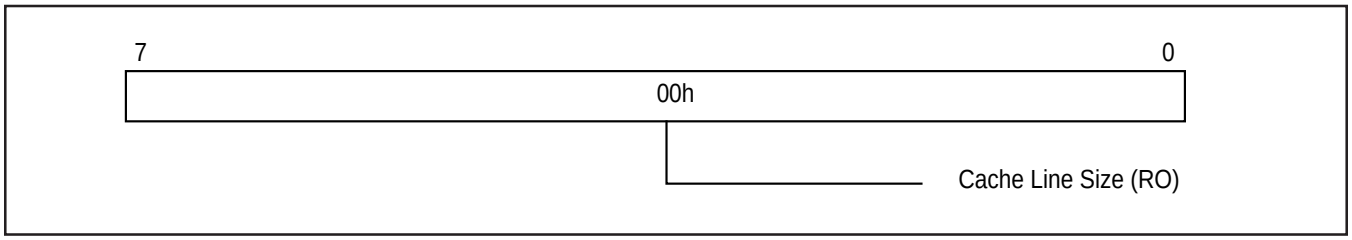
Sub-Class	Prog I/F	Description
00	00h	FireWire™ (IEEE 1394)
01h	00h	ACCESS.bus
02h	00h	SSA

CACHE LINE SIZE REGISTER (CALN)

Register Name:	Cache Line Size
Address Offset:	0Ch
Power-up value:	00h, hardwired
Boot-load:	not used
Attribute:	Read Only
Size:	8 bits

This register is hardwired to 0. The cache line configuration register is used by the system to define the cache line size in doubleword (64-bit) increments. This controller does not use the “Memory Write and Invalidate” PCI bus cycle commands when operating in the bus master mode, and therefore does not internally require this register. When operating in the target mode, this controller does not have the connections necessary to “snoop” the PCI bus and accordingly cannot employ this register in the detection of burst transfers that cross a line boundary.

Figure 7. Cache Line Size Register

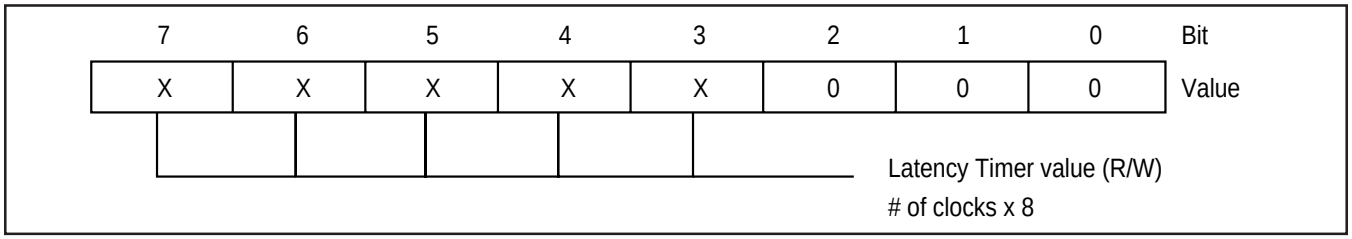


LATENCY TIMER REGISTER (LAT)

Register Name:	Latency Timer
Address Offset:	0Dh
Power-up value:	00h
Boot-load:	External nvRAM offset 04Dh
Attribute:	Read/Write, bits 7:3; Read Only bits 2:0
Size:	8 bits

The latency timer register has meaning only when this controller is used as a bus master and pertains to the number of PCI bus clocks that this master will be guaranteed. The nonzero value for this register is internally decremented after this device has been granted the bus and has begun to assert FRAME#. Prior to this latency timer count reaching zero, this device can ignore the removal of the bus grant and may continue the use of the bus for data transfers.

Figure 8. Latency Timer Register

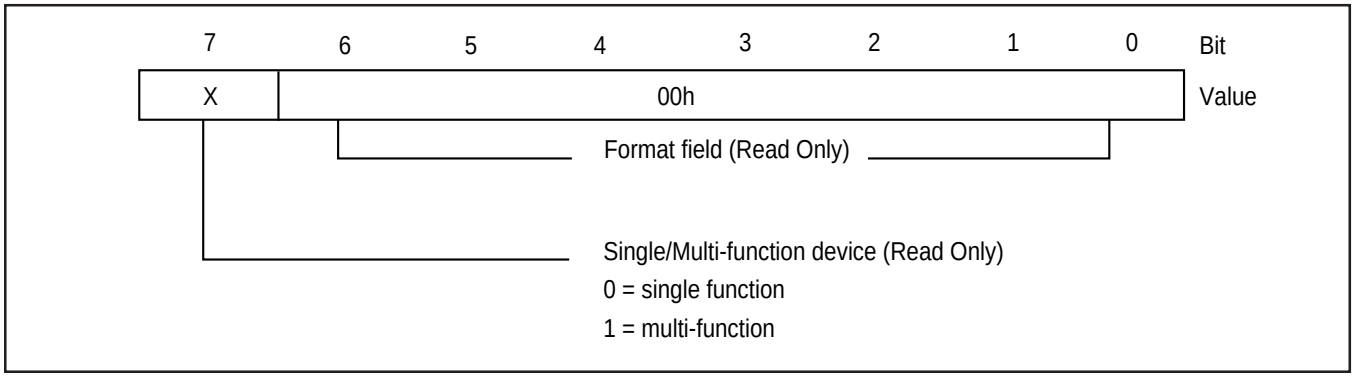


HEADER TYPE REGISTER (HDR)

Register Name: Header Type
Address Offset: 0Eh
Power-up value: 00h
Boot-load: External nvRAM offset 04Eh
Attribute: Read Only
Size: 8 bits

This register consists of two fields: Bits 6:0 define the format for bytes 10h through 3Fh of the device configuration header, and bit 7 establishes whether this device represents a single function (bit 7 = 0) or a multifunction (bit 7 = 1) PCI bus agent. The S5933 is a single function PCI device.

Figure 9. Header Type Register



BUILT-IN SELF-TEST REGISTER (BIST)

Register Name: Built-in Self-Test
Address Offset: 0Fh
Power-up value: 00h
Boot-load: External nvRAM/EPROM offset 04Fh
Attribute: D7, D5-0 Read Only, D6 as PCI bus write only
Size: 8 bits

The Built-In Self-Test (BIST) register permits the implementation of custom, user-specific diagnostics. This register has four fields as depicted in Figure 10. Bit 7, when set signifies that this device supports a built-in self test. When bit 7 is set, writing a 1 to bit 6 will commence the self test. In actuality, writing a 1 to bit 6 produces an interrupt to the Add-On interface. Bit 6 will remain set until cleared by a write operation to this register from the Add-On bus interface. When bit 6 is reset it is interpreted as completion of the self-test and an error is indicated by a non-zero value for the completion code (bits 3:0).

Figure 10. Built-In Self Test Register

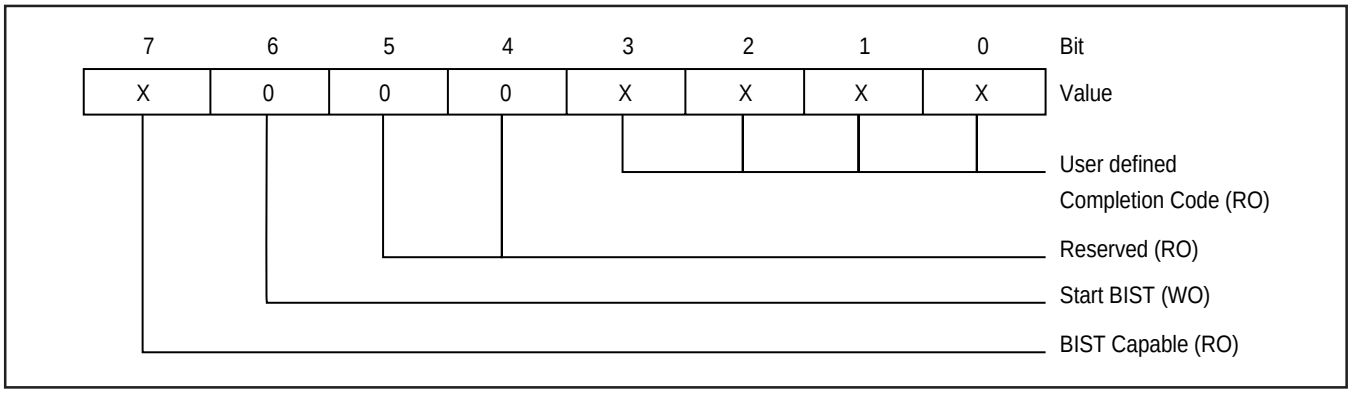


Table 21. Built-In Self-Test Register

Bit	Description
7	BIST Capable. This bit indicates that the Add-On device supports a built-in self-test when a one is returned. A zero should be returned if this self test feature is not desired. This field is read only from the PCI interface.
6	Start BIST. Writing a 1 to this bit indicates that the self-test should commence. This bit can only be written when bit 7 is a 1. When bit 6 becomes set, an interrupt is issued to the Add-On interface. Other than through the reset pin, Bit 6 can only be cleared by a write to this element from the Add-On bus interface as outlined in Section 6.5. The PCI bus specification requires that this bit be cleared within 2 seconds after being set, or the device will be failed.
5:4	Reserved. These bits are reserved. This field will always return zeros.
3:0	Completion Code. This field provides a method for detailing a device-specific error. It is considered valid when the Start BIST field (bit 6) changes from 1 to 0. An all-zero value for the completion code indicates successful completion.

BASE ADDRESS REGISTERS (BADR)

Register Name:	Base Address
Address Offset:	10h, 14h, 18h, 1Ch, 20h, 24h
Power-up value:	FFFFFFC1h for offset 10h; 00000000h for all others
Boot-load:	External nvRAM offset 050h, 54h, 58h, 5Ch, 60h (BADR0-4)
Attribute:	high bits Read/Write; low bits Read Only
Size:	32 bits

The base address registers provide a mechanism for assigning memory or I/O space for the Add-On function. The actual location(s) the Add-On function is to respond to is determined by first interrogating these registers to ascertain the size or space desired, and then writing the high-order field of each register to place it physically in the system's address space. Bit zero of each field is used to select whether the space required is to be decoded as memory (bit 0 = 0) or I/O (bit 0 = 1). Since this PCI controller has 16 DWORDs of internal operating registers, the Base Address Register at offset 10h is assigned to them. The remaining five base address registers can only be used by boot-loading them from the external nvRAM interface. BADR5 register is not implemented and will return all 0's.

Determining Base Address Size

The address space defined by a given base address register is determined by writing all 1s to a given base address register from the PCI bus and then reading that register back. The number of 0s returned starting from D4 for memory space and D2 for I/O space toward the high-order bits reveals the amount of address space desired. Tables 23 and 24 list the possible returned values and their corresponding size for both memory and I/O, respectively. Included in the table are the nvRAM/EPROM boot values which correspond to a given assigned size. A register returning all zeros is disabled.

Assigning the Base Address

After a base address has been sized as described in the preceding paragraph, the region associated with that base address register (the high order one bits) can physically locate it in memory (or I/O) space. For example, the first base address register returns FFFFFFC1h indicating an I/O space (D0=1) and is then written with the value 00000300h. This means that the controller's internal registers can be selected for I/O addresses between 00000300h through 0000033Fh, in this example. The base address value must be on a natural binary boundary for the required size (example 300h, 340h, 380h etc.; 338h would not be allowable).

Figure 11a. Base Address Register — Memory

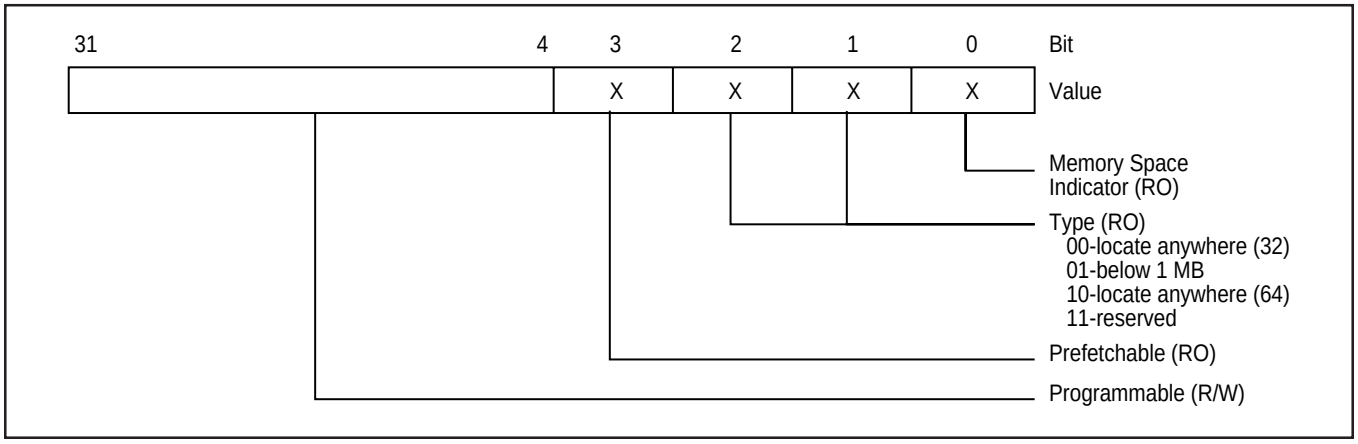


Figure 11b. Base Address Register — I/O

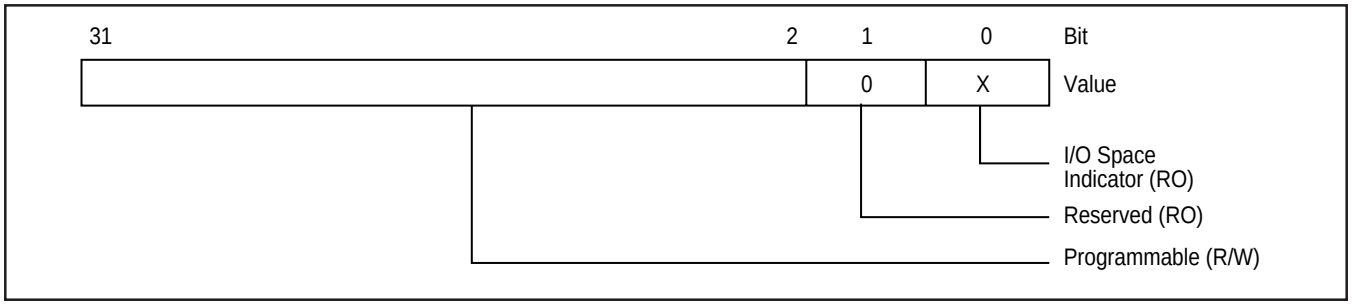


Table 22a. Base Address Register — Memory (Bit 0 = 0)

Bit	Description
31:4	Base Address Location. These bits are used to position the decoded region in memory space. Only bits which return a 1 after being written as 1 are usable for this purpose. Except for Base Address Register 0, these bits are individually enabled by the contents sourced from the external boot memory.
3	Prefetchable. When set as a 1, this bit signifies that this region of memory can be cached. Cachable regions can only be located within the region altered through PCI bus memory writes. This bit, when set, also implies that all read operations will return the data associated for all bytes regardless of the Byte Enables. Memory space which cannot support this behavior should leave this bit in the zero state. For Base Addresses 1 through 4, this bit is set by the Reset pin and later initialized by the external boot memory (if present). Base Address Register 0 always has this bit set to 0. This bit is read only from the PCI interface.
2:1	Memory Type. These two bits identify whether the memory space is 32 or 64 bits wide and if the space location is restricted to be within the first megabyte of memory space. The table below describes the encoding: **Bits 2 1** — **Description** 0 0 — Region is 32 bits wide and can be located anywhere in 32 bit memory space. 0 1 — Region is 32 bits wide and must be mapped below the first MByte of memory space. 1 0 — Region is 64 bits wide and can be mapped anywhere within 64 bit memory space. (Not supported by this controller.) 1 1 — Reserved. (Not supported by this controller.)
1	The 64-bit memory space is not supported by this controller, so bit 2 should not be set. The only meaningful option is whether it is desired to position memory space anywhere within 32-bit memory space or restrain it to the first megabyte. For Base Addresses 1 through 5, this bit is set by the reset pin and later initialized by the external boot memory (if present).
0	Space Indicator = 0. When set to 0, this bit identifies a base address region as a memory space and the remaining bits in the base address register are defined as shown in Table 22a.

Table 22b. Base Address Register — I/O (Bit 0 = 1)

Bit	Description
31:2	Base Address Location. These bits are used to position the decoded region in I/O space. Only bits which return a “1” after being written as “1” are usable for this purpose. Except for Base Address 0, these bits are individually enabled by the contents sourced from the external boot memory (EPROM or nvRAM).
1	Reserved. This bit should be zero. (Note: disabled Base Address Registers will return all zeros for the entire register location, bits 31 through 0).
0	Space Indicator = 1. When one this bit identifies a base address region as an I/O space and the remaining bits in the base address register have the definition as shown in Table 11b.

Table 23. Read Response (Memory Assigned) to an All-Ones Write Operation to a Base Address Register

Response	Size in bytes	[EPROM boot value] [1]
00000000h	none - disabled	00000000h or BIOS missing [2,3]
FFFFFFF0h	16 bytes (4 DWORDs)	FFFFFFF0h
FFFFFFE0h	32 bytes (8 DWORDs)	FFFFFFE0h
FFFFFFC0h	64 bytes (16 DWORDs)	FFFFFFC0h
FFFFFF80h	128 bytes (32 DWORDs)	FFFFFF80h
FFFFFF00h	256 bytes (64 DWORDs)	FFFFFF00h
FFFFFE00h	512 bytes (128 DWORDs)	FFFFFE00h
FFFFFC00h	1K bytes (256 DWORDs)	FFFFFC00h
FFFFF800h	2K bytes (512 DWORDs)	FFFFF800h
FFFFF000h	4K bytes (1K DWORDs)	FFFFF000h
FFFFE000h	8K bytes (2K DWORDs)	FFFFE000h
FFFFC000h	16K bytes (4K DWORDs)	FFFFC000h
FFFF8000h	32K bytes (8K DWORDs)	FFFF8000h
FFFF0000h	64K bytes (16K DWORDs)	FFFF0000h
FFFE0000h	128K bytes (32K DWORDs)	FFFE0000h
FFFC0000h	256K bytes (64K DWORDs)	FFFC0000h
FFF80000h	512K bytes (128K DWORDs)	FFF80000h
FFF00000h	1M bytes (256K DWORDs)	FFF00000h
FFE00000h	2M bytes (512K DWORDs)	FFE00000h
FFC00000h	4M bytes (1M DWORDs)	FFC00000h
FF800000h	8M bytes (2M DWORDs)	FF800000h
FF000000h	16M bytes (4M DWORDs)	FF000000h
FE000000h	32M bytes (8M DWORDs)	FE000000h
FC000000h	64M bytes (16M DWORDs)	FC000000h
F8000000h	128M bytes (32M DWORDs)	F8000000h
F0000000h	256M bytes (64M DWORDs)	F0000000h
E0000000h	512M bytes (128M DWORDs)	E0000000h

1. The two most significant bits define bus width for BADR1:4 in Pass-Thru operation).
2. Bits D3, D2 and D1 may be set to indicate other attributes for the memory space. See text for details.
3. BADR5 register is not implemented and will return all 0's.

Table 24. Read Response (I/O Assigned) to an All-Ones write Operation to a Base Address Register

Response	Size in bytes	[EPROM boot value]
00000000h	none - disabled	00000000h or BIOS missing [3]
FFFFFFFDh	4 bytes (1 DWORDs)	FFFFFFFDh
FFFFFFF9h	8 bytes (2 DWORDs)	FFFFFFF9h
FFFFFFF1h	16 bytes (4 DWORDs)	FFFFFFF1h
FFFFFFE1h	32 bytes (8 DWORDs)	FFFFFFE1h
FFFFFFC1h	64 bytes (16 DWORDs)	FFFFFFC1h [4]
FFFFFF81h	128 bytes (32 DWORDs)	FFFFFF81h
FFFFFF01h	256 bytes (64 DWORDs)	FFFFFF01h

4. Base Address Register 0 (at offset) 10h powers up as FFFFFFC1h. This default assignment allows usage without an external boot memory. Should an EPROM or nvRAM be used, the base address can be boot loaded to become a memory space (FFFFFFC0h or FFFFFFC2h).

EXPANSION ROM BASE ADDRESS REGISTER (XROM)

Register Name:	Expansion ROM Base Address
Address Offset:	30h
Power-up value:	00000000h
Boot-load:	External nvRAM offset 70h
Attribute:	bits 31:11, bit 0 Read/Write; bits 10:1 Read Only
Size:	32 bits

The expansion base address ROM register provides a mechanism for assigning a space within physical memory for an expansion ROM. Access from the PCI bus to the memory space defined by this register will cause one or more accesses to the S5933 controllers' external BIOS ROM (or nvRAM) interface. Since PCI bus accesses to the ROM may be 32 bits wide, repeated operations to the ROM are generated by the S5933 and the wider data is assembled internal to the S5933 controller and then transferred to the PCI bus by the S5933.

Figure 12. Expansion ROM Base Address Register

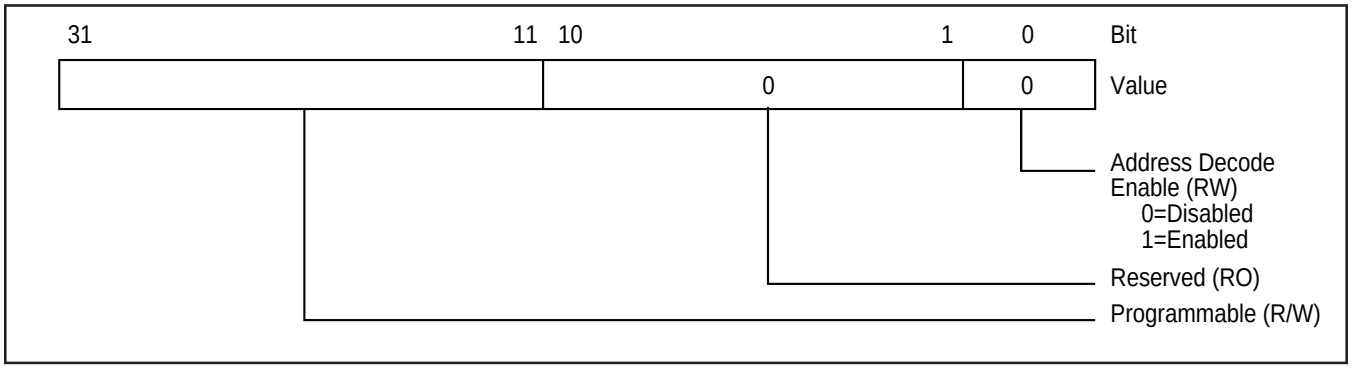


Table 25. Expansion ROM Base Address Register

Bit	Description
31:11	Expansion ROM Base Address Location. These bits are used to position the decoded region in memory space. Only bits which return a 1 after being written as 1 are usable for this purpose. These bits are individually enabled by the contents sourced from the external boot memory (EPROM or nvRAM). The desired size for the ROM memory is determined by writing all ones to this register and then reading back the contents. The number of bits returned as zeros, in order from least significant to most significant bit, indicates the size of the expansion ROM. This controller limits the expansion ROM area to 64K bytes. The allowable returned values after all ones are written to this register are shown in Table 26.
10:1	Reserved. All zeros.
0	Address Decode Enable. The Expansion ROM address decoder is enabled or disabled with this bit. When this bit is set, the decoder is enabled; when this bit is zero, the decoder is disabled. It is required that the PCI command register also have the memory decode enabled for this bit to have an effect.

Table 26. Read Response to Expansion ROM Base Address Register (after all-ones written)

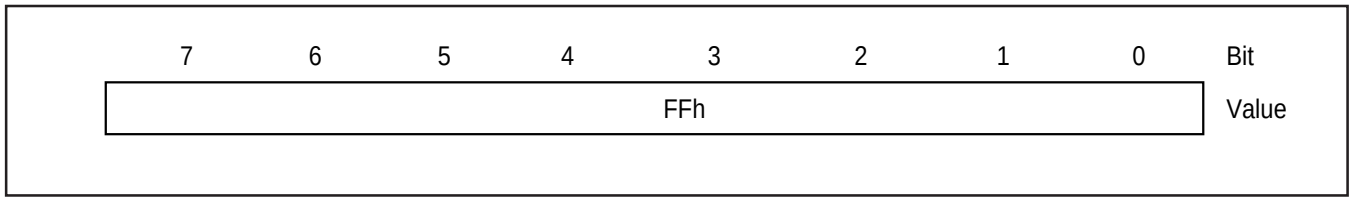
Response	Size in bytes	[EPROM boot value]
00000000h	none - disabled	00000000h or BIOS missing
FFFFF801h	2K bytes (512 DWORDs)	FFFFF801h
FFFFF001h	4K bytes (1K DWORDs)	FFFFF001h
FFFFE001h	8K bytes (2K DWORDs)	FFFFE001h
FFFFC001h	16K bytes (4K DWORDs)	FFFFC001h
FFFF8001h	32K bytes (8K DWORDs)	FFFF8001h
FFFF0001h	64K bytes (16K DWORDs)	FFFF0001h

INTERRUPT LINE REGISTER (INTLN)

Register Name:	Interrupt Line
Address Offset:	3Ch
Power-up value:	FFh
Boot-load:	External nvRAM offset 7Ch
Attribute:	Read/Write
Size:	8 bit

This register indicates the interrupt routing for the S5933 controller. The ultimate value for this register is system-architecture specific. For x86 based PCs, the values in this register correspond with the established interrupt numbers associated with the dual 8259 controllers used in those machines. In x86-based PC systems, the values of 0 to 15 correspond with the IRQ numbers 0 through 15, and the values from 16 to 254 are reserved. The value of 255 (the controller's default power-up value) signifies either "unknown" or "no connection" for the system interrupt. This register is boot-loaded from the external boot memory, if present, and may be written by the PCI interface.

Figure 13. Interrupt Line Register

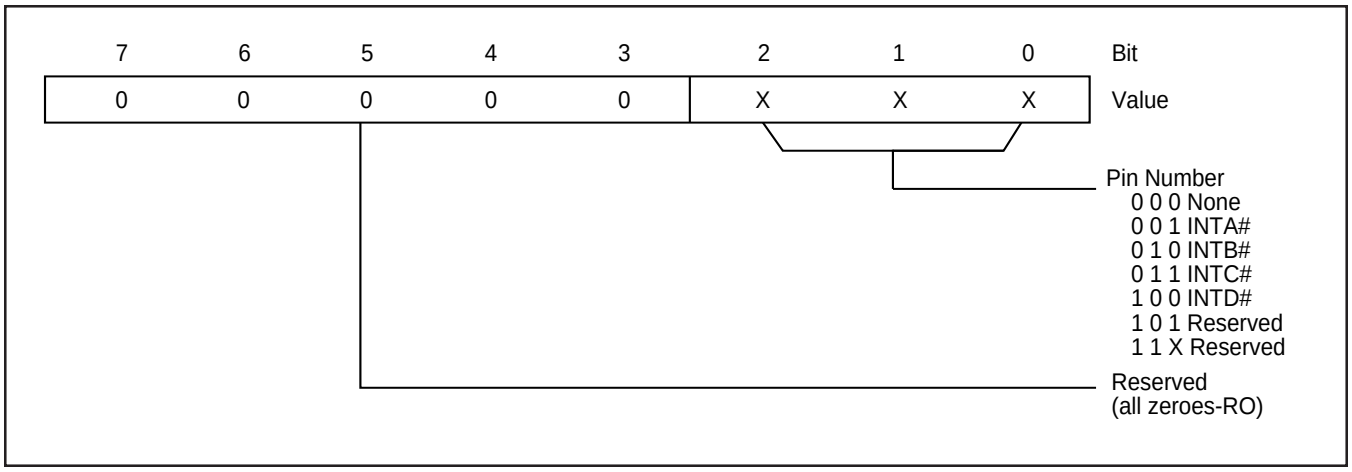


INTERRUPT PIN REGISTER (INTPIN)

Register Name: Interrupt Pin
Address Offset: 3Dh
Power-up value: 01h
Boot-load: External nvRAM offset 7Dh
Attribute: Read Only
Size: 8 bits

This register identifies which PCI interrupt, if any, is connected to the controller's PCI interrupt pins. The allowable values are 0 (no interrupts), 1 (INTA#), 2 (INTB#), 3 (INTC#), and 4 (INTD#). The default power-up value assumes INTA#.

Figure 14. Interrupt Pin Register



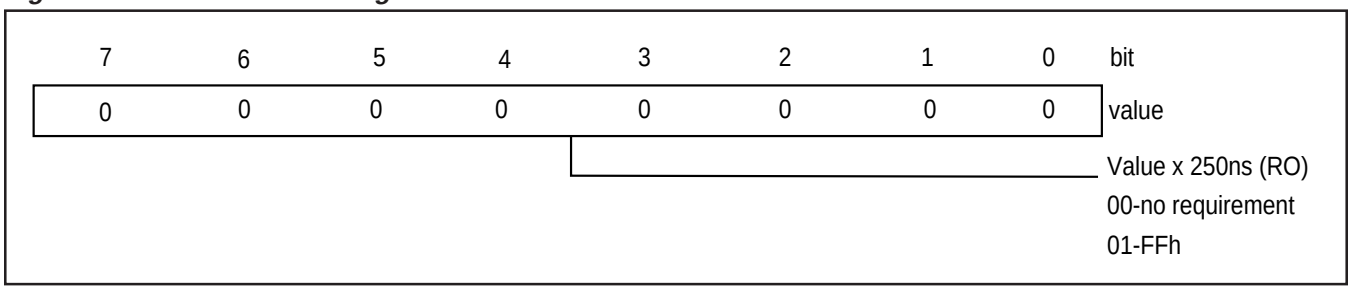
MINIMUM GRANT REGISTER (MINGNT)

Register Name:	Minimum Grant
Address Offset:	3Eh
Power-up value:	00h
Boot-load:	External nvRAM offset 7Eh
Attribute:	Read Only
Size:	8 bits

This register may be optionally used by bus masters to specify how long a burst period the device needs. A value of zero indicates that the bus master has no stringent requirement. The units defined by the least significant bit are in 250-ns increments. This register is treated as “information only” and has no further implementation within this device.

Values other than zero are possible when an external boot memory is used.

Figure 15. Minimum Grant Register



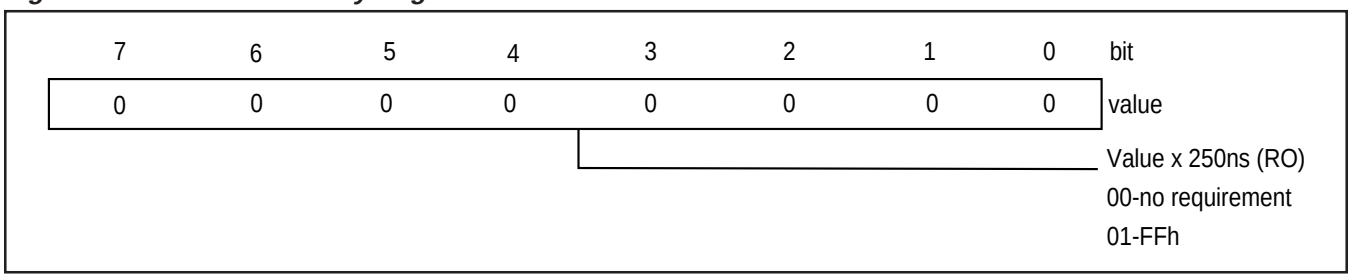
MAXIMUM LATENCY REGISTER (MAXLAT)

Register Name: Maximum Latency
Address Offset: 3Fh
Power-up value: 00h
Boot-load: External nvRAM offset 7Fh
Attribute: Read Only
Size: 8 bits

This register may be optionally used by bus masters to specify how often this device needs PCI bus access. A value of zero indicates that the bus master has no stringent requirement. The units defined by the least significant bit are in 250-ns increments. This register is treated as “information only” and has no further implementation within this device.

Values other than zero are possible when an external boot memory is used.

Figure 16. Maximum Latency Register



PCI BUS OPERATION REGISTERS

The PCI bus operation registers are mapped as 16 consecutive DWORD registers located at the address space (I/O or memory) specified by the Base Address Register 0. These locations are the primary method of communication between the PCI and Add-On buses. Data, software-defined commands and command parameters can be either exchanged through the mailboxes, transferred through the FIFO in blocks under program control, or transferred using the FIFOs under Bus Master control. Table 1 lists the PCI Bus Operation Registers.

Table 1. Operation Registers — PCI Bus

Address Offset	Abbreviation	Register Name
00h	OMB1	Outgoing Mailbox Register 1
04h	OMB2	Outgoing Mailbox Register 2
08h	OMB3	Outgoing Mailbox Register 3
0Ch	OMB4	Outgoing Mailbox Register 4
10h	IMB1	Incoming Mailbox Register 1
14h	IMB2	Incoming Mailbox Register 2
18h	IMB3	Incoming Mailbox Register 3
1Ch	IMB4	Incoming Mailbox Register 4
20h	FIFO	FIFO Register port (bidirectional)
24h	MWAR	Master Write Address Register
28h	MWTC	Master Write Transfer Count Register
2Ch	MRAR	Master Read Address Register
30h	MRTC	Master Read Transfer Count Register
34h	MBEF	Mailbox Empty/Full Status
38h	INTCSR	Interrupt Control/Status Register
3Ch	MCSR	Bus Master Control/Status Register

OUTGOING MAILBOX REGISTERS (OMB)

Register Names: Outgoing Mailboxes 1-4

PCI Address Offset: 00h, 04h, 08h, 0Ch

Power-up value: XXXXXXXXh

Attribute: Read/Write

Size: 32 bits

These four DWORD registers provide a method for sending command or parameter data to the Add-On system. PCI bus operations to these registers may be in any width (byte, word, or DWORD). Writing to these registers can be a source for Add-On bus interrupts (if desired) by enabling their interrupt generation through the use of the Add-On's interrupt control/status register.

INCOMING MAILBOX REGISTERS (IMB)

Register Names: Incoming Mailboxes 1-4

PCI Address Offset: 10h, 14h, 18h, 1Ch

Power-up value: XXXXXXXXh

Attribute: Read Only

Size: 32 bits

These four DWORD registers provide a method for receiving user defined data from the Add-On system. PCI bus read operations to these registers may be in any width (byte, word, or DWORD). Only read operations are supported. Reading from these registers can optionally cause an Add-On bus interrupt (if desired) by enabling their interrupt generation through the use of the Add-On's interrupt control/status register.

Mailbox 4, byte 3 only exists as device pins on the S5933 devices when used with a serial nonvolatile memory.

FIFO REGISTER PORT (FIFO)

Register Name: FIFO Port

PCI Address Offset: 20h

Power-up value: XXXXXXXXh

Attribute: Read/Write

Size: 32 bits

This location provides access to the bidirectional FIFO. Separate registers are used when reading from or writing to the FIFO. Accordingly, it is not possible to read what was written to this location. The FIFO registers are implicitly involved in all bus master operations and, as such, should not be accessed during active bus master transfers. When operating upon the FIFOs with software program transfers involving word or byte operations, the *endian* sequence of the FIFO should be established as described under FIFO Endian Conversion Management in order to preserve the internal FIFO data ordering and flag management. The FIFO's fullness may be observed by reading the master control- status registeror MCSR register.

PCI CONTROLLED BUS MASTER WRITE ADDRESS REGISTER (MWAR)

Register Name: Master Write Address
PCI Address Offset: 24h
Power-up value: 00000000h
Attribute: Read/Write
Size: 32 bits

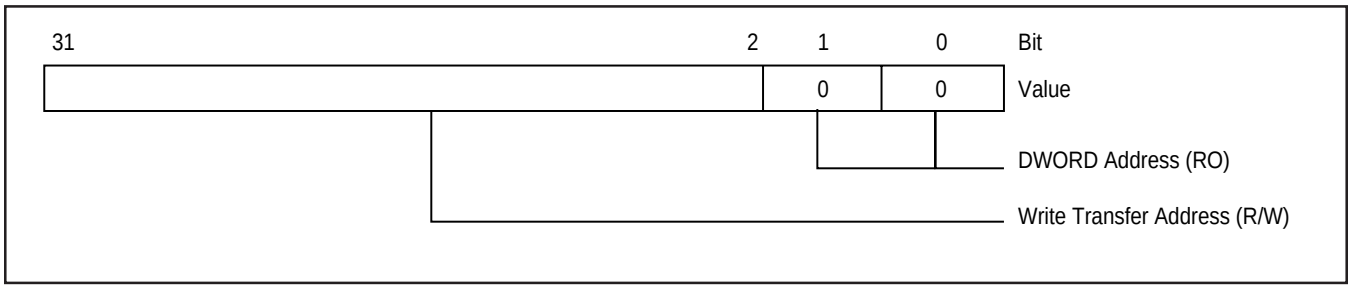
This register is used to establish the PCI address for data moving from the Add-On bus to the PCI bus during PCI bus memory write operations. It consists of a 30-bit counter with the low-order two bits hardwired as zeros. Transfers may be any non-zero byte length as defined by the transfer count register, MWTC, and must begin on a DWORD boundary. This DWORD boundary starting constraint is placed upon this controller's PCI bus master transfers so that byte lane alignment can be maintained between the S5933 controller's internal FIFO data path, the Add-On interface, and the PCI bus.

Note: Applications which require a non-DWORD starting boundary will need to move the first few bytes under software program control (and without using the FIFO) to establish a DWORD boundary. After the DWORD boundary is established the S5933 can begin the task of PCI bus master data transfers.

The Master Write Address Register is continually updated during the transfer process and will always be pointing to the next unwritten location. Reading of this register during a transfer process (done when the S5933 controller is functioning as a target, i.e. not a bus master) is permitted and may be used to monitor the progress of the transfer. During the address phase for bus master write transfers, the two least significant bits presented on the PCI bus pins AD[31:0] will always be zero. This identifies to the target memory that the burst address sequence will be in a linear order rather than in an Intel 486 or Pentium™ cache line fill sequence. Also, the PCI bus address bit A1 will always be zero when this controller is the bus master. This signifies to the target that the S5933 controller is burst capable and that the target should not arbitrarily disconnect after the first data phase of this operation.

Under certain circumstances, MWAR can be accessed from the Add-On bus instead of the PCI bus. See Add-On Initiated Bus Mastering.

Figure 1. PCI Controlled Bus Master Write Address Register



PCI CONTROLLED BUS MASTER WRITE TRANSFER COUNT REGISTER (MWTC)

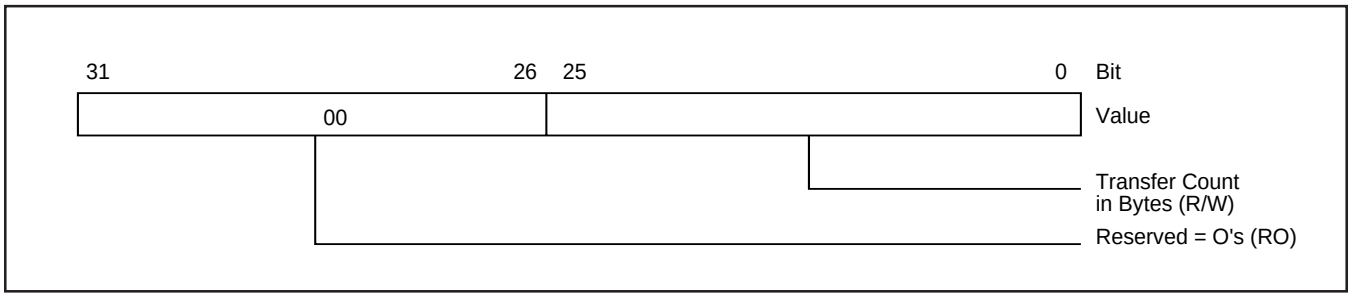
Register Name: Master Write Transfer Count
PCI Address Offset: 28h
Power-up value: 00000000h
Attribute: Read/Write
Size: 32 bits

The master write transfer count register is used to convey to the S5933 controller the actual number of bytes that are to be transferred. The value in this register is decremented with each bus master PCI write operation until the transfer count reaches zero.

Upon reaching zero, the transfer operation ceases and an interrupt may be optionally generated to either the PCI or Add-On bus interface. Transfers which are not whole multiples of DWORDs in size result in a partial word ending cycle. This partial word ending cycle is possible since all bus master transfers for this controller are required to begin on a DWORD boundary.

Under certain circumstances, MWTC can be accessed from the Add-On bus instead of the PCI bus. See Add-On Initiated Bus Mastering.

Figure 2. PCI Controlled Bus Master Write Transfer Count Register



PCI CONTROLLED BUS MASTER READ ADDRESS REGISTER (MRAR)

Register Name: Master Read Address
PCI Address Offset: 2Ch
Power-up value: 00000000h
Attribute: Read/Write
Size: 32 bits

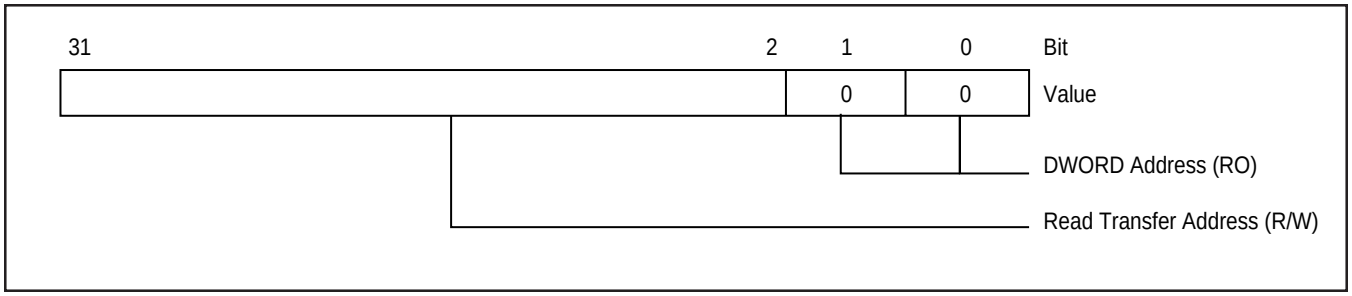
This register is used to establish the PCI address for data moving to the Add-On bus from the PCI bus during PCI bus memory read operations. It consists of a 30-bit counter with the low-order two bits hardwired as zeros. Transfers may be any non-zero byte length as defined by the transfer count register, MRTC (Section 5.7) and must begin on a DWORD boundary. This DWORD boundary starting constraint is placed upon this controller's PCI bus master transfers so that byte lane alignment can be maintained between the S5933 controller's internal FIFO data path, the Add-On interface and the PCI bus.

Note: Applications which require a non-DWORD starting boundary will need to move the first few bytes under software program control (and without using the FIFO) to establish a DWORD boundary. After the DWORD boundary is established the S5933 can begin the task of PCI bus master data transfers.

The Master Read Address Register is continually updated during the transfer process and will always be pointing to the next unread location. Reading of this register during a transfer process (done when the S5933 controller is functioning as a target—i.e., not a bus master) is permitted and may be used to monitor the progress of the transfer. During the address phase for bus master read transfers, the two least significant bits presented on the PCI bus AD[31:0] will always be zero. This identifies to the target memory that the burst address sequence will be in a linear order rather than in an Intel 486 or Pentium™ cache line fill sequence. Also, the PCI bus address bit A1 will always be zero when this controller is the bus master. This signifies to the target that the controller is burst capable and that the target should not arbitrarily disconnect after the first data phase of this operation.

Under certain circumstances, MRAR can be accessed from the Add-On bus instead of the PCI bus.

Figure 3. PCI Controlled Bus Master Read Address Register



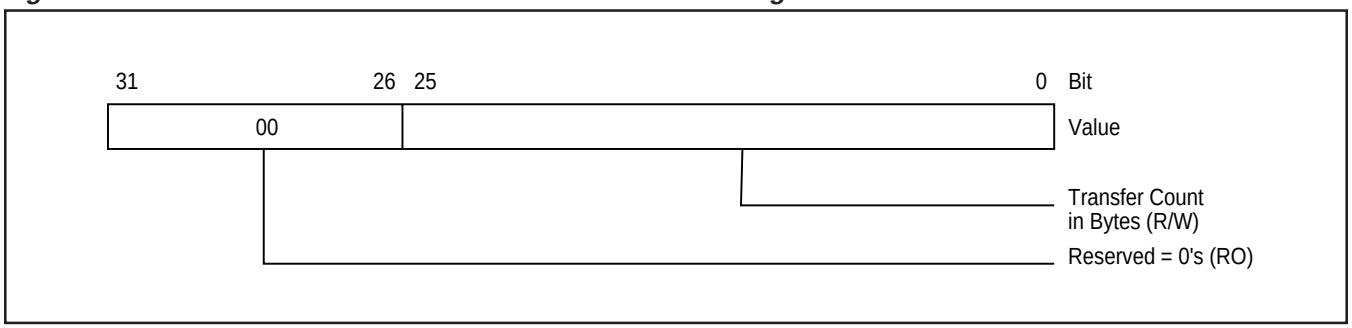
PCI CONTROLLED BUS MASTER READ TRANSFER COUNT REGISTER (MRTC)

Register Name: Master Read Transfer Count
PCI Address Offset: 30h
Power-up value: 00000000h
Attribute: Read/Write
Size: 32 bits

The master read transfer count register is used to convey to the PCI controller the actual number of bytes that are to be transferred. The value in this register is decremented with each bus master PCI read operation until the transfer count reaches zero. Upon reaching zero, the transfer operation ceases and an interrupt may be optionally generated to either the PCI or Add-On bus interface. Transfers which are not whole multiples of DWORDs in size result in a partial word ending cycle. This partial word ending cycle is possible since all bus master transfers for this controller are required to begin on a DWORD boundary.

Under certain circumstances, MRTC can be accessed from the Add-On bus instead of the PCI bus.

Figure 4. PCI Controlled Bus Master Read Transfer Count Register



MAILBOX EMPTY FULL/STATUS REGISTER (MBEF)

Register Name: Mailbox Empty/Full Status

PCI Address Offset: 34h

Power-up value: 00000000h

Attribute: Read Only

Size: 32 bits

This register provides empty/full visibility of each byte within the mailboxes. The empty/full status for the Outgoing mailboxes is displayed on the low-order 16 bits and the empty/full status for the Incoming mailboxes is presented on the high-order 16 bits. A value of 1 signifies that a given mailbox has been written by one bus interface but has not yet been read by the corresponding destination interface. A PCI bus incoming mailbox is defined as one in which data travels from the Add-On bus into the PCI bus, and an outgoing mailbox is defined as one where data travels out from the PCI bus to the Add-On interface.

Figure 5. Mailbox Empty/Full Status Register

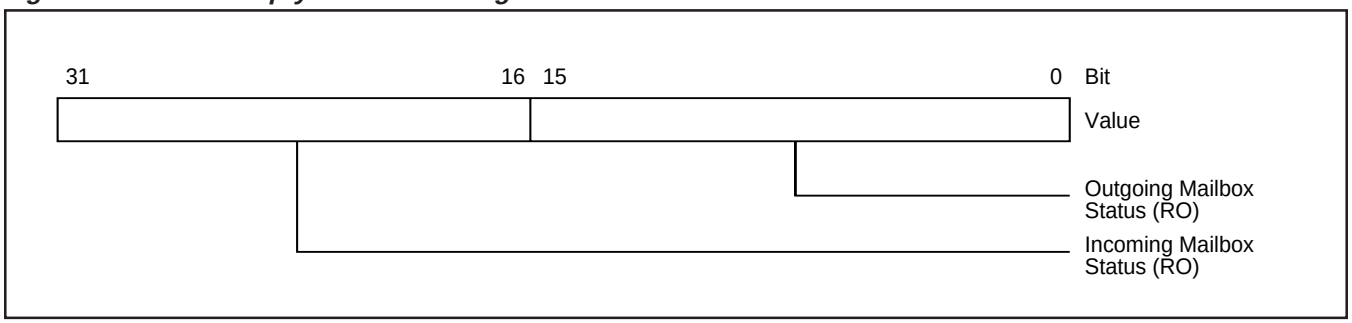


Table 2. Mailbox Empty/Full Status Register

Bit	Description
31:16	Incoming Mailbox Status. This field indicates which incoming mailbox registers have been written by the Add-On interface but have not yet been read by the PCI bus. Each bit location corresponds to a specific byte within one of the four incoming mailboxes. A value of one for each bit signifies that the specified mailbox byte is full, and a value of zero signifies empty. The mapping of these status bits to bytes within each mailbox is as follows: Bit 31 = Incoming mailbox 4 byte 3 Bit 30 = Incoming mailbox 4 byte 2 Bit 29 = Incoming mailbox 4 byte 1 Bit 28 = Incoming mailbox 4 byte 0 Bit 27 = Incoming mailbox 3 byte 3 Bit 26 = Incoming mailbox 3 byte 2 Bit 25 = Incoming mailbox 3 byte 1 Bit 24 = Incoming mailbox 3 byte 0 Bit 23 = Incoming mailbox 2 byte 3 Bit 22 = Incoming mailbox 2 byte 2 Bit 21 = Incoming mailbox 2 byte 1 Bit 20 = Incoming mailbox 2 byte 0 Bit 19 = Incoming mailbox 1 byte 3 Bit 18 = Incoming mailbox 1 byte 2 Bit 17 = Incoming mailbox 1 byte 1 Bit 16 = Incoming mailbox 1 byte 0
15:00	Outgoing Mailbox Status. This field indicates which out going mail box registers have been written by the PCI bus interface but have not yet been read by the Add-On bus. Each bit location corresponds to a specific byte within one of the four outgoing mailboxes. A value of one for each bit signifies that the specified mailbox byte is full, and a value of zero signifies empty. The mapping of these status bits to bytes within each mailbox is as follows: Bit 15 = Outgoing mailbox 4 byte 3 Bit 14 = Outgoing mailbox 4 byte 2 Bit 13 = Outgoing mailbox 4 byte 1 Bit 12 = Outgoing mailbox 4 byte 0 Bit 11 = Outgoing mailbox 3 byte 3 Bit 10 = Outgoing mailbox 3 byte 2 Bit 09 = Outgoing mailbox 3 byte 1 Bit 08 = Outgoing mailbox 3 byte 0 Bit 07 = Outgoing mailbox 2 byte 3 Bit 06 = Outgoing mailbox 2 byte 2 Bit 05 = Outgoing mailbox 2 byte 1 Bit 04 = Outgoing Mailbox 2 byte 0 Bit 03 = Outgoing Mailbox 1 byte 3 Bit 02 = Outgoing Mailbox 1 byte 2 Bit 01 = Outgoing Mailbox 1 byte 1 Bit 00 = Outgoing Mailbox 1 byte 0

INTERRUPT CONTROL/STATUS REGISTER (INTCSR)

Register Name: Interrupt Control and Status
PCI Address Offset: 38h
Power-up value: 00000000h
Attribute: Read/Write (R/W), Read/Write_One_Clear (R/WC)
Size: 32 bits

This register provides the method for choosing which conditions are to produce an interrupt on the PCI bus interface, a method for viewing the cause of the interrupt, and a method for acknowledging (removing) the interrupt's assertion.

Interrupt sources:

- Write Transfer Terminal Count = zero
- Read Transfer Terminal Count = zero
- One of the Outgoing mailboxes (1,2,3 or 4) becomes empty
- One of the Incoming mailboxes (1,2,3 or 4) becomes full.
- Target Abort
- Master Abort

Figure 6. Interrupt Control/Status Register

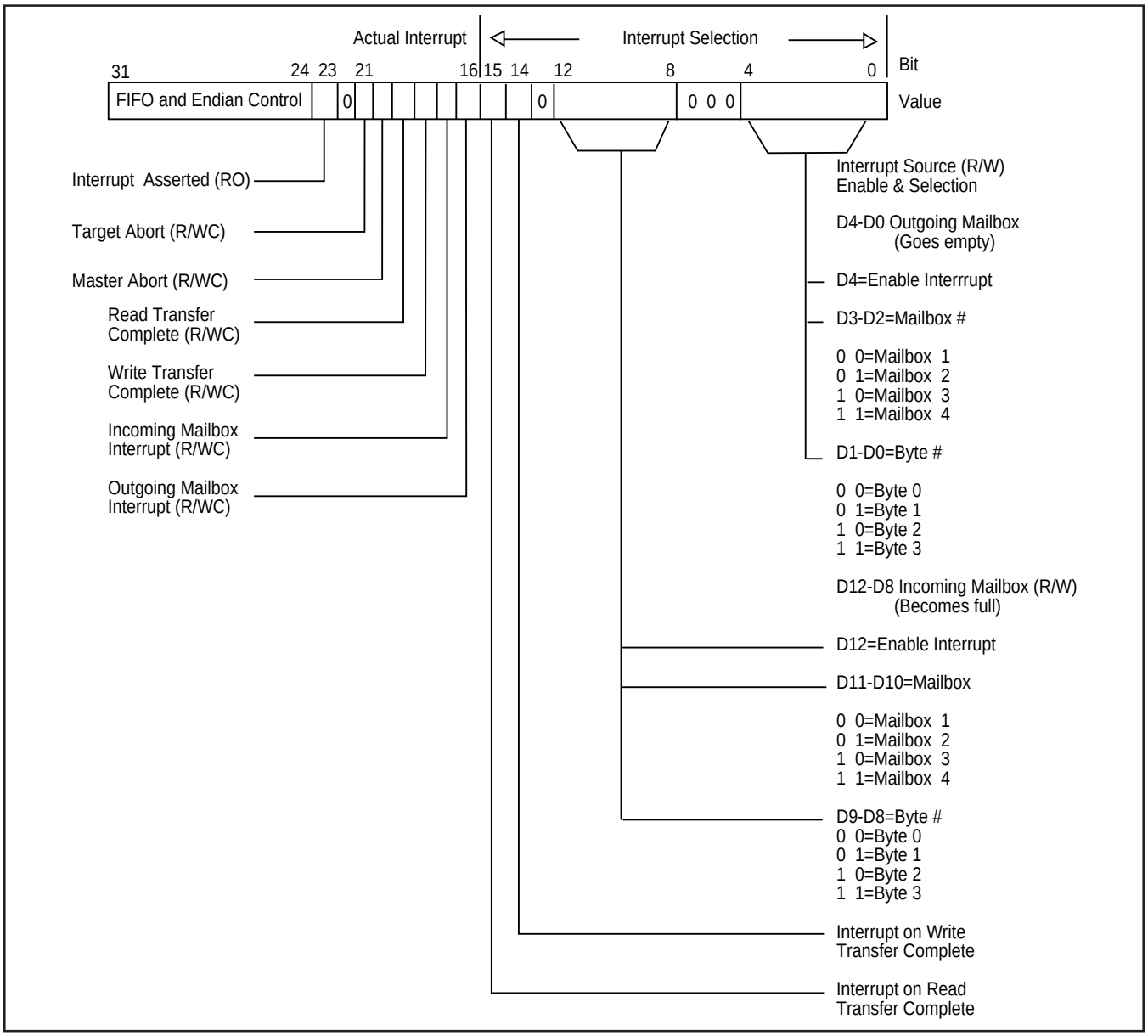


Figure 7. FIFO Management and Endian Control Byte

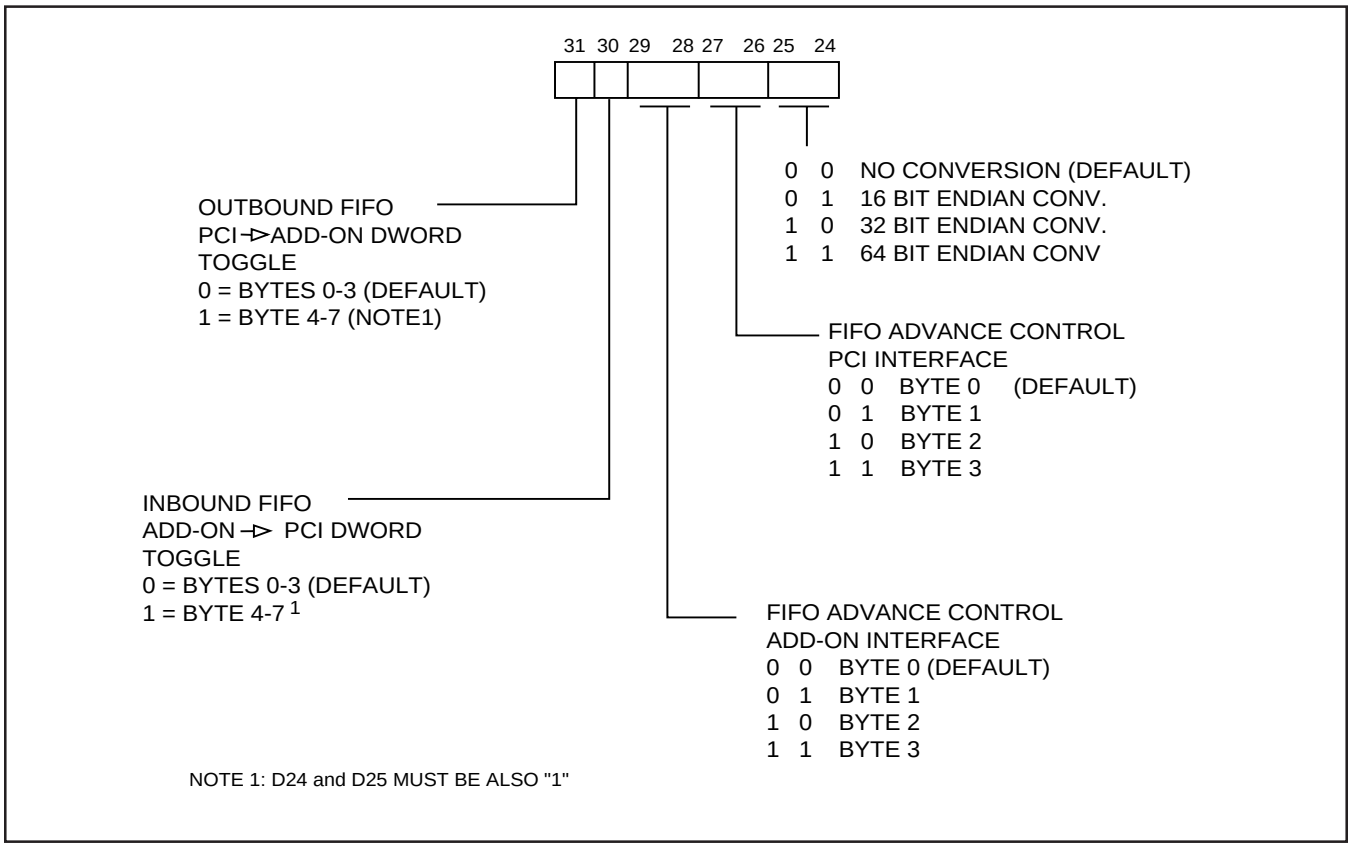


Table 3. Interrupt Control/Status Register

Bit	Description
31:24	FIFO and Endian Control.
23	Interrupt asserted. This read only status bit indicates that one or more of the four possible interrupt conditions is present. This bit is nothing more than the ORing of the interrupt conditions described by bits 19 through 16 of this register.
22	Reserved. Always zero.
21	Target Abort. This bit signifies that an interrupt has been generated due to the S5933 encountering a target abort during a PCI bus cycle while the S5933 was the current bus master. This bit operates as read or write one clear. A write to this bit with the data of “one” will cause this bit to be reset, a write to this bit with the data of “zero” will not change the state of this bit.
20	Master Abort. This bit signifies that an interrupt has been generated due to the S5933 encountering a Master Abort on the PCI bus. A master abort occurs when there is no target response to a PCI bus cycle. This bit operates as read or write one clear. A write to this bit with the data of “one” will cause this bit be reset, a write to this bit with the data of “zero” will not change the state of this bit.
19	Read Transfer Complete. This bit signifies that an interrupt has been generated due to the completion of a PCI bus master operation involving the transfer of data from the PCI bus to the Add-On. This interrupt will occur when the Master Read Transfer Count register reaches zero. This bit operates as read or write one clear. A write to this bit with the data of “one” will cause this bit to be reset; a write to this bit with the data of “zero” will not change the state of this bit.
18	Write Transfer Complete. This bit signifies that an interrupt has been generated due to the completion of a PCI bus master operation involving the transfer of data to the PCI bus from the Add-On. This interrupt will occur when the Master Write Transfer Count register reaches zero. This bit operates as read or write one clear. A write to this bit with the data of “one” will cause this bit to be reset; a write to this bit with the data of “zero” will not change the state of this bit.
17	Incoming Mailbox Interrupt. This bit is set when the mailbox selected by bits 12 through 8 of this register are written by the Add-On interface. This bit operates as read or write one clear. A write to this bit with the data of “one” will cause this bit to be reset; a write to this bit with the data as “zero” will not change the state of this bit.
16	Outgoing Mailbox Interrupt. This bit is set when the mailbox selected by bits 4 through 0 of this register is read by the Add-On interface. This bit operates as read or write one clear. A write to this bit with the data of “one” will cause this bit to be reset; a write to this bit with the data of “zero” will not change the state of this bit.
15	Interrupt on Read Transfer Complete. This bit enables the occurrence of an interrupt when the read transfer count reaches zero. This bit is read/write.
14	Interrupt on Write Transfer Complete. This bit enables the occurrence of an interrupt when the write transfer count reaches zero. This bit is read/write.
13	Reserved. Always zero.
12	Enable incoming mailbox interrupt. This bit allows a write from the incoming mailbox register identified by bits 11 through 8 to produce a PCI interface interrupt. This bit is read/write.
11:10	Incoming Mailbox Interrupt Select. This field selects which of the four incoming mailboxes is to be the source for causing an incoming mailbox interrupt. [00]b selects mailbox 1, [01]b selects mailbox 2, [10]b selects mailbox 3 and [11]b selects mailbox 4. This field is read/write.

Table 3. Interrupt Control/Status Register (Continued)

Bit	Description
9:8	Incoming Mailbox Byte Interrupt select. This field selects which byte of the mailbox selected by bits 10 and 11 above is to actually cause the interrupt. [00]b selects byte 0, [01]b selects byte 1, [10]b selects byte 2, and [11]b selects byte 3. This field is read/write.
7:5	Reserved, Always zero.
4	Enable outgoing mailbox interrupt. This bit allows a read by the Add-On of the outgoing mailbox register identified by bits 3 through 0 to produce a PCI interface interrupt. This bit is read/write.
3:2	Outgoing Mailbox Interrupt Select. This field selects which of the four outgoing mailboxes is to be the source for causing an outgoing mailbox interrupt. [00]b selects mailbox 1, [01]b selects mailbox 2, [10]b selects mailbox 3 and [11]b selects mailbox 4. This field is read/write.
1:0	Outgoing Mailbox Byte Interrupt select. This field selects which byte of the mailbox selected by bits 3 and 2 above is to actually cause the interrupt. [00]b selects byte 0, [01]b selects byte 1, [10]b selects byte 2, and [11]b selects byte 3. This field is read/write.

MASTER CONTROL/STATUS REGISTER (MCSR)

Register Name: Master Control/Status
PCI Address Offset: 3Ch
Power-up value: 000000E6h
Attribute: Read/Write, Read Only, Write Only
Size: 32 bits

This register provides for overall control of this device. It is used to enable bus mastering for both data directions as well as providing a method to perform software resets.

The following PCI bus controls are available:

- Write Priority over Read
- Read Priority over Write
- Write Transfer Enable
- Write master requests on 4 or more FIFO words available (full)
- Read transfer enable
- Read master requests on 4 or more FIFO available (empty)
- Assert reset to Add-On
- Reset Add-On to PCI FIFO flags
- Reset PCI to Add-On FIFO flags
- Reset mailbox empty full status flags
- Write external non-volatile memory

The following PCI interface status flags are provided:

- PCI to Add-On FIFO FULL
- PCI to Add-On FIFO has four or more empty locations
- PCI to Add-On FIFO EMPTY
- Add-On to PCI FIFO FULL
- Add-On to PCI FIFO has four or more words loaded
- Add-On to PCI FIFO EMPTY
- PCI to Add-On Transfer Count = Zero
- Add-On to PCI Transfer Count = Zero

Figure 8. Bus Master Control/Status Register

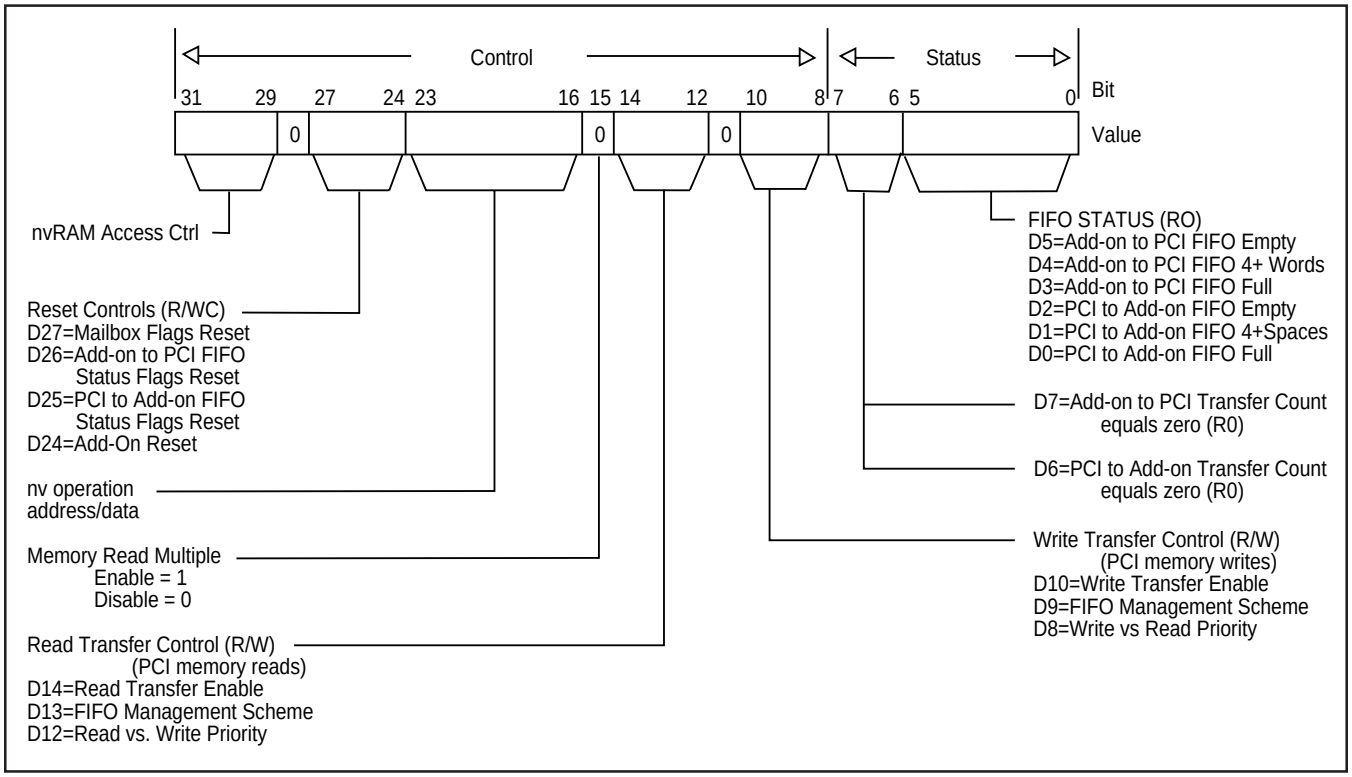


Table 4. Bus Master Control/Status Register

Bit	Description
31:29	nvRAM Access Control. This field provides a method for access to the optional external non-volatile memory. Write operations are achieved by a sequence of byte operations involving these bits and the 8-bit field of bits 23 through 16. The sequence requires that the low-order address, high order address, and then a data byte are loaded in order. Bit 31 of this field acts as a combined enable and ready for the access to the external memory. D31 must be written to a 1 before an access can begin, and subsequent accesses must wait for bit D31 to become zero (ready). D31 D30 D29 W/R 0 X X W Inactive 1 0 0 W Load low address byte 1 0 1 W Load high address byte 1 1 0 W Begin write 1 1 1 W Begin read 0 X X R Ready 1 X X R Busy Cautionary note: The nonvolatile memory interface is also available for access by the Add-On interface. Accesses by both the Add-On and PCI bus to the nv memory are not directly supported by the S5933 device. Software must be designed to prevent the simultaneous access of nv memory to prevent data corruption within the memory and provide for accurate data retrieval.
28	FIFO loop back mode.
27	Mailbox Flag Reset. Writing a one to this bit causes all mailbox status flags to become reset (EMPTY). It is not necessary to write this bit as zero because it is used internally to produce a reset pulse. Since reading of this bit will always produce zeros, this bit is write only.
26	Add-On to PCI FIFO Status Reset. Writing a one to this bit causes the Add-On to PCI (Bus master memory writes) FIFO empty flag to set indicating empty and the FIFO FULL flag to reset and the FIFO Four Plus word flag to reset. It is not necessary to write this bit as zero because it is used internally to produce a reset pulse. Since reading of this bit will always produce zeros, this bit is write only.
25	PCI to Add-On FIFO Status Reset. Writing a one to this bit causes the PCI to Add-On (Bus master memory reads) FIFO empty flag to set indicating empty and the FIFO FULL flag to reset and the FIFO Four Plus words available flag to set. It is not necessary to write this bit as zero because it is used internally to produce a reset pulse. Since reading of this bit will always produce zeros, this bit is write only.
24	Add-On pin reset. Writing a one to this bit causes the reset output pin to become active. Writing a zero to this pin is necessary to remove the assertion of reset. This register bit is read/write.
23:16	Non-volatile memory address/data port. This 8-bit field is used in conjunction with bit 31, 30 and 29 of this register to access the external non-volatile memory. The contents written are either low address, high address, or data as defined by bits 30 and 29. This register will contain the external non-volatile memory data when the proper read sequence for bits 31 through 29 is performed.

Table 4. Bus Master Control/Status Register (Continued)

Bit	Description
15	Enable memory read multiple during S5933 bus mastering mode.
14	Read Transfer Enable. This bit must be set to a one for S5933 PCI bus master read transfers to take place. Writing a zero to this location will suspend an active transfer. An active transfer is one in which the transfer count is not zero.
13	Read FIFO management scheme. When set to a 1, this bit causes the controller to refrain from requesting the PCI bus unless it has four or more vacant FIFO locations to fill. Once the controller is granted the PCI bus or is in possession of the bus due to the write channel, this constraint is not meaningful. When this bit is zero the controller will request the PCI bus if it has at least one vacant FIFO word.
12	Read versus Write priority. This bit controls the priority of read transfers over write transfers. When set to a 1 with bit D8 as zero this indicates that read transfers always have priority over write transfers; when set to a one with D8 as one, this indicates that transfer priorities will alternate equally between read and writes.
11	Reserved. Always zero.
10	Write Transfer Enable. This bit must be set to a one for PCI bus master write transfers to take place. Writing a zero to this location will suspend an active transfer. An active transfer is one in which the transfer count is not zero.
9	Write FIFO management scheme. When set to a one this bit causes the controller to refrain from requesting the PCI bus unless it has four or more FIFO locations filled. Once the S5933 controller is granted the PCI bus or is in possession of the bus due to the write channel, this constraint is not meaningful. When this bit is zero the controller will request the PCI bus if it has at least one valid FIFO word.
8	Write versus Read priority. This bit controls the priority of write transfers over read transfers. When set to a one with bit D12 as zero this indicates that write transfers always have priority over read transfers; when set to a one with D12 as one, this indicates that transfer priorities will alternate equally between writes and reads.
7	Add-On to PCI Transfer Count Equal Zero (RO). This bit is a one to signify that the write transfer count is all zeros.
6	PCI to Add-On Transfer Count Equals Zero (RO). This bit is a one to signify that the read transfer count is all zeros.
5	Add-On to PCI FIFO Empty. This bit is a one when the Add-On to PCI bus FIFO is completely empty.
4	Add-On to PCI 4+ words. This bit is a one when there are four or more FIFO words valid within the Add-On to PCI bus FIFO.
3	Add-On to PCI FIFO Full. This bit is a one when the Add-On to PCI bus FIFO is completely full.
2	PCI to Add-On FIFO Empty. This bit is a one when the PCI bus to Add-On FIFO is completely empty.
1	PCI to Add-On FIFO 4+ spaces. This bit signifies that there are at least four empty words within the PCI to Add-On FIFO.
0	PCI to Add-On FIFO Full. This bit is a one when the PCI bus to Add-On FIFO is completely full.

ADD-ON BUS OPERATION REGISTERS

The Add-On bus interface provides access to 18 DWORDs (72 bytes) of data, control and status information. All of these locations are accessed by asserting the Add-On bus chip select pin (SELECT#) in conjunction with either the read or write control strobes (signal pin RD# or WR#). Access to the FIFO can also be achieved through use of the dedicated pins, RDFIFO# and WRFIFO#. The dedicated pins for control of the FIFO are provided to optionally implement Direct Memory Access (DMA) on the Add-On bus, or to connect with an external FIFO.

This register group represents the primary method for communication between the Add-On and PCI buses as viewed by the Add-On. The flexibility of this arrangement allows a number of user-defined software protocols to be built. For example, data, software assigned commands, and command parameters can be exchanged between the PCI and Add-On buses using either the mailboxes or FIFOs with or without handshaking interrupts. The register structure is very similar to that of the PCI operation register set. The major difference between the PCI bus and Add-On bus register complement are the absence of bus master control registers (4) on the Add-On side and the addition of two “pass-through” registers. Table 1 lists the Add-On interface registers.

Table 1. Operation Registers — Add-On Interface

Address	Abbreviation	Register Name
00h	AIMB1	Add-On Incoming Mailbox Register #1
04h	AIMB2	Add-On Incoming Mailbox Register #2
08h	AIMB3	Add-On Incoming Mailbox Register #3
0Ch	AIMB4	Add-On Incoming Mailbox Register #4
10h	AOMB1	Add-On Outgoing Mailbox Register #1
14h	AOMB2	Add-On Outgoing Mailbox Register #2
18h	AOMB3	Add-On Outgoing Mailbox Register #3
1Ch	AOMB4	Add-On Outgoing Mailbox Register #4
20h	AFIFO	Add-On FIFO port
24h	MWAR [1]	Bus Master Write Address Register
28h	APTA	Add-On Pass-Through Address
2Ch	APTD	Add-On Pass-Through Data
30h	MRAR [1]	Bus Master Read Address Register
34h	AMBEF	Add-On Mailbox Empty/Full Status
38h	AINT	Add-On Interrupt control
3Ch	AGCSTS	Add-On General Control and Status Register
58h	MWTC [1]	Bus Master Write Transfer Count
5Ch	MRTC [1]	Bus Master Read Transfer Count

1. See Add-On Initiated Bus Mastering.

ADD-ON INCOMING MAILBOX REGISTERS (AIMBx)

Register Names: Add-On Incoming Mailboxes 1-4

Add-On Address Offset: 00h, 04h, 08h, 0Ch

Power-up value: XXXXXXXXh

Attribute: Read Only

Size: 32 bits

These four DWORD registers provide a method for receiving data, commands, or command parameters from the PCI interface. Add-On read operations to these registers may be in any width (byte, word, or DWORD). These registers are read-only. Writes to this address space have no effect. Reading from one of these registers can optionally cause a PCI bus interrupt (if desired) when the PCI interrupt control/status register is properly configured.

ADD-ON OUTGOING MAILBOX REGISTERS (AOMBx)

Register Names: Add-On Outgoing Mailboxes 1-4

Add-On Address Offset: 10h, 14h, 18h, 1Ch

Power-up value: XXXXXXXXh

Attribute: Read/Write

Size: 32 bits

These four DWORD registers provide a method for sending data, commands, or command parameters or status to the PCI interface. Add-On write operations to these registers may be in any width (byte, word, or DWORD). These registers may also be read. Writing to one of these registers can optionally cause a PCI bus interrupt (if desired) when the PCI interrupt control/status register is properly configured. Mailbox 4, byte 3 only exists as device pins on the S5933 device when used with a serial nonvolatile memory. This byte is not available if a byte-wide nv memory is used.

ADD-ON FIFO REGISTER PORT (AFIFO)

Register Name: Add-On FIFO Port

Add-On Address Offset: 20h

Power-up value: XXXXXXXXh

Attribute: Read/Write

Size: 32 bits

This location provides access to the bidirectional FIFO. Separate registers are involved when reading and writing to this location. Accordingly, it is not possible to read what was written to this location. The sequence of filling and emptying this FIFO is established by the PCI interface interrupt control and Status Register.

The FIFO's fullness may be observed by reading the master control/status register or AGCSTS register Additionally, two signal pins are provided which reveal whether data is available (RDEMPTY) or space to write into the FIFO is available (WRFULL). These signals may be used to interface with user supplied DMA logic. Caution must be exercised when using these flags for FIFO transfers involving 64 bit endian conversion since the FIFO must operate on DWORD pairs.

ADD-ON CONTROLLED BUS MASTER WRITE ADDRESS REGISTER (MWAR)

Register Name: Master Write Address

Add-On Address Offset: 24h

Power-up value: 00000000h

Attribute: Read/Write

Size: 32 bits

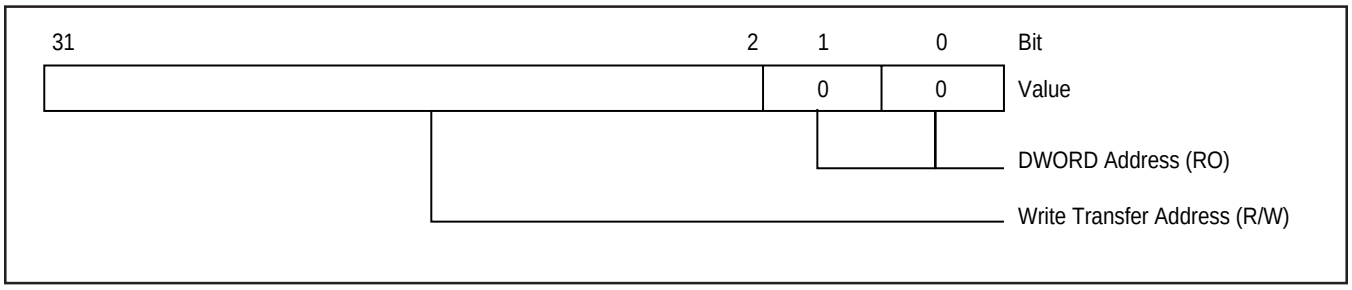
This register is only accessible when Add-On initiated bus mastering is enabled.

This register is used to establish the PCI address for data moving from the Add-On bus to the PCI bus during PCI bus memory write operations. It consists of a 30-bit counter with the low-order two bits hardwired as zeros. Transfers may be any non-zero byte length as defined by the transfer count register, MWTC and must begin on a DWORD boundary. This DWORD boundary starting constraint is placed upon this controller's PCI bus master transfers so that byte lane alignment can be maintained between the S5933 controller's internal FIFO data path, the Add-On interface, and the PCI bus.

Note: Applications which require a non-DWORD starting boundary will need to move the first few bytes under software program control (and without using the FIFO) to establish a DWORD boundary. After the DWORD boundary is established the S5933 can begin the task of PCI bus master data transfers.

The Master Write Address Register is continually updated during the transfer process and will always be pointing to the next unwritten location. Reading of this register during a transfer process (done when the S5933 controller is functioning as a target, i.e. not a bus master) is permitted and may be used to monitor the progress of the transfer. During the address phase for bus master write transfers, the two least significant bits presented on the PCI bus pins AD[31:0] will always be zero. This identifies to the target memory that the burst address sequence will be in a linear order rather than in an Intel 486 or Pentium™ cache line fill sequence. Also, the PCI bus address bit A1 will always be zero when this controller is the bus master. This signifies to the target that the S5933 controller is burst capable and that the target should not arbitrarily disconnect after the first data phase of this operation.

Figure 1. Add-On Controlled Bus Master Write Address Register



ADD-ON PASS-THRU ADDRESS REGISTER (APTA)

Register Name: Add-On Pass-Thru Address

Add-On Address Offset: 28h

Power-up value: XXXXXXXXh

Attribute: Read Only

Size: 32 bits

This register is employed when a response is desired when one of the Base address decode regions is selected during an active PCI bus cycle. When one of the base address decode registers 1-4 encounters a PCI bus cycle which selects the region defined by it, this device latches that current cycle's active address and asserts the signal PTATN# (Pass-Thru Attention). Wait states are generated on the PCI bus until either data is transferred or the PCI bus cycle is aborted by the initiator.

This register provides a method for "live" data (registered) transfers. Intended uses include the emulating of other hardware as well as enabling the connection of existing external hardware to interface to the PCI bus through the S5933.

ADD-ON PASS-THRU DATA REGISTER (APTD)

Register Name: Add-On Pass-Thru Data

Add-On Address Offset: 2Ch

Power-up value: XXXXXXXXh

Attribute: Read/Write

Size: 32 bits

This register, along with APTA described above, is employed when a response is desired should one of the Base address decode regions become selected during an active PCI bus cycle. When one of the base address decode registers 1-4 encounters a PCI bus cycle which selects the region defined by it, the APTA register will contain that current cycle's active address and the device asserts the signal PTATN# (Pass-Thru ATentioN). Wait states are generated on the PCI bus until this register is read (PCI bus writes) or this register is written (PCI bus reads).

ADD-ON CONTROLLED BUS MASTER READ ADDRESS REGISTER (MRAR)

Register Name: Master Read Address

Add-On Address Offset: 30h

Power-up value: 00000000h

Attribute: Read/Write

Size: 32 bits

This register is only accessible when Add-On initiated bus mastering is enabled.

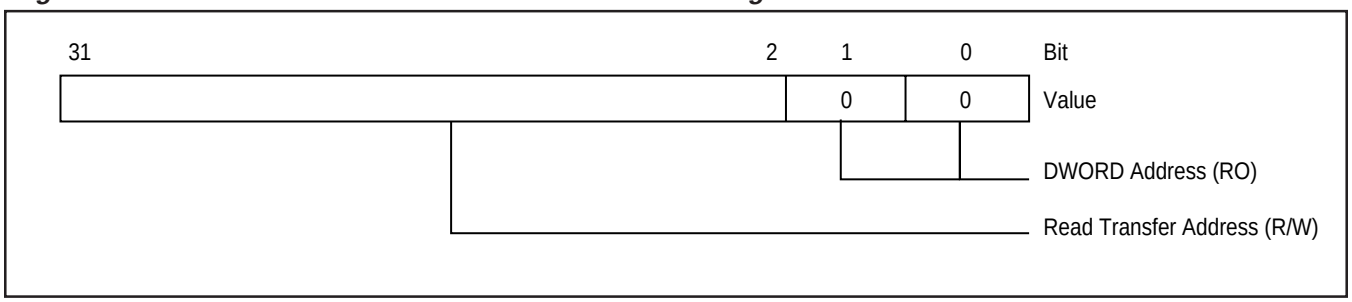
This register is used to establish the PCI address for data moving to the Add-On bus from the PCI bus during PCI bus memory read operations. It consists of a 30-bit counter with the low-order two bits hardwired as zeros. Transfers may be any non-zero byte length as defined by the transfer count register, MRTC and must begin on a DWORD boundary. This DWORD boundary starting constraint is placed upon this controller's PCI bus master transfers so that byte lane alignment can be maintained between the S5395X controller's internal FIFO data path, the Add-On interface and the PCI bus.

Note: Applications which require a non-DWORD starting boundary will need to move the first few bytes under software program control (and without using the FIFO) to establish a DWORD boundary. After the DWORD boundary is established the S5933 can begin the task of PCI bus master data transfers.

The Master Read Address Register is continually updated during the transfer process and will always be pointing to the next unread location. Reading of this register during a transfer process (done when the S5933 controller is functioning as a target—i.e., not a bus master) is permitted and may be used to monitor the progress of the transfer. During the address phase for bus master read transfers, the two least significant bits presented on the PCI bus AD[31:0] will always be zero. This identifies to the target memory that the burst address sequence will be in a linear order rather than in an Intel 486 or Pentium™ cache line fill sequence. Also, the PCI bus address bit A1 will always be zero when this controller is the bus master. This signifies to the target that the controller is burst capable and that the target should not arbitrarily disconnect after the first data phase of this operation.

Under certain circumstances, MRAR can be accessed from the Add-On bus instead of the PCI bus.

Figure 2. Add-On Controlled Bus Master Read Address Register



ADD-ON EMPTY/FULL STATUS REGISTER (AMBEF)

Register Name: Add-On Mailbox Empty/Full Status

Add-On Address Offset: 34h

Power-up value: 00000000h

Attribute: Read Only

Size: 32 bits

This register provides empty/full visibility of each byte within the mailboxes. The empty/full status for the Outgoing mailboxes are displayed on the high order 16 bits and the empty/full status for the incoming mailboxes are presented on the low order 16 bits. A value of one signifies that a given mailbox had been written by the sourcing interface but had not yet been read by the corresponding destination interface. An incoming mailbox is defined as one in which data travels from the PCI bus into the Add-On bus and an outgoing mailbox is defined as one where data goes OUT from the Add-On bus to the PCI interface.

Figure 3. Add-On Mailbox Empty/Full Status Register

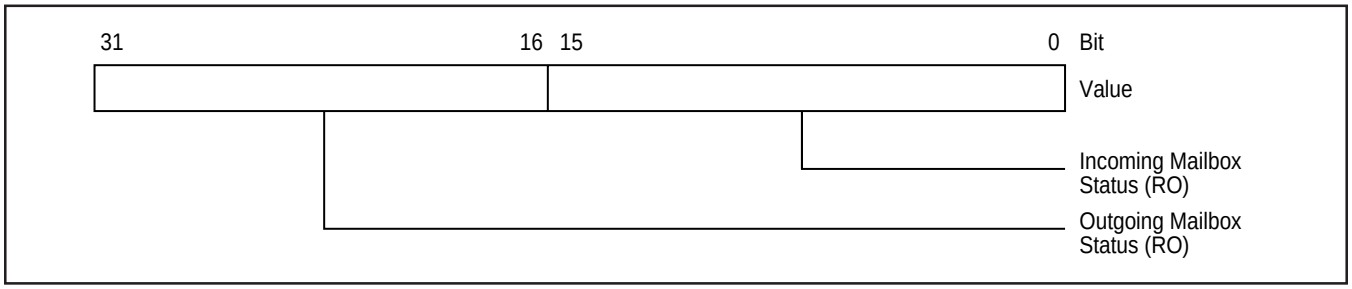


Table 2. Add-On Mailbox Empty/Full Status Register

Bit	Description
31:16	Outgoing Mailbox Status. This field indicates which outgoing mailbox registers have been written by the Add-On bus interface but have not yet been read by the PCI bus. Each bit location corresponds to a specific byte within one of the four outgoing mailboxes. A value of one for each bit signifies that the specified mailbox byte is full, a value of zero signifies empty. The mapping of these status bits to bytes within each mailbox is as follows: Bit 31 = Outgoing mailbox 4 byte 3 Bit 30 = Outgoing mailbox 4 byte 2 Bit 29 = Outgoing mailbox 4 byte 1 Bit 28 = Outgoing mailbox 4 byte 0 Bit 27 = Outgoing mailbox 3 byte 3 Bit 26 = Outgoing mailbox 3 byte 2 Bit 25 = Outgoing mailbox 3 byte 1 Bit 24 = Outgoing mailbox 3 byte 0 Bit 23 = Outgoing mailbox 2 byte 3 Bit 22 = Outgoing mailbox 2 byte 2 Bit 21 = Outgoing mailbox 2 byte 1 Bit 20 = Outgoing mailbox 2 byte 0 Bit 19 = Outgoing mailbox 1 byte 3 Bit 18 = Outgoing mailbox 1 byte 2 Bit 17 = Outgoing mailbox 1 byte 1 Bit 16 = Outgoing mailbox 1 byte 0
15:00	Incoming Mailbox Status. This field indicates which incoming mailbox registers have been written by the PCI bus but not yet been read by the Add-On interface. Each bit location corresponds to a specific byte within one of the four incoming mailboxes. A value of one for each bit signifies that the specified mailbox byte is full, a value of zero signifies empty. The mapping of these status bits to bytes within each mailbox is as follows: Bit 15 = Incoming mailbox 4 byte 3 Bit 14 = Incoming mailbox 4 byte 2 Bit 13 = Incoming mailbox 4 byte 1 Bit 12 = Incoming mailbox 4 byte 0 Bit 11 = Incoming mailbox 3 byte 3 Bit 10 = Incoming mailbox 3 byte 2 Bit 9 = Incoming mailbox 3 byte 1 Bit 8 = Incoming mailbox 3 byte 0 Bit 7 = Incoming mailbox 2 byte 3 Bit 6 = Incoming mailbox 2 byte 2 Bit 5 = Incoming mailbox 2 byte 1 Bit 4 = Incoming mailbox 2 byte 0 Bit 3 = Incoming mailbox 1 byte 3 Bit 2 = Incoming mailbox 1 byte 2 Bit 1 = Incoming mailbox 1 byte 1 Bit 0 = Incoming mailbox 1 byte 0

ADD-ON INTERRUPT CONTROL/ STATUS REGISTER (AINT)

Register Name: Add-On Interrupt Control and Status

Add-On Address Offset: 38h

Power-up value: 00000000h

Attribute: Read/Write, Read/Write_One_Clear

Size: 32 bits

This register provides the method for choosing which conditions are to produce an interrupt on the Add-On bus interface, a method for viewing the cause for the interrupt, and a method for acknowledging (removing) the interrupt's assertion.

Interrupt sources:

- One of the Incoming mailboxes (1,2,3 or 4) becomes full.
- One of the Outgoing mailboxes (1,2,3 or 4) becomes empty.
- Built-in self test issued.
- Write Transfer Count = zero
- Read Transfer Count = zero
- Target/Master Abort

Figure 4. Add-On Interrupt Control/Status Register

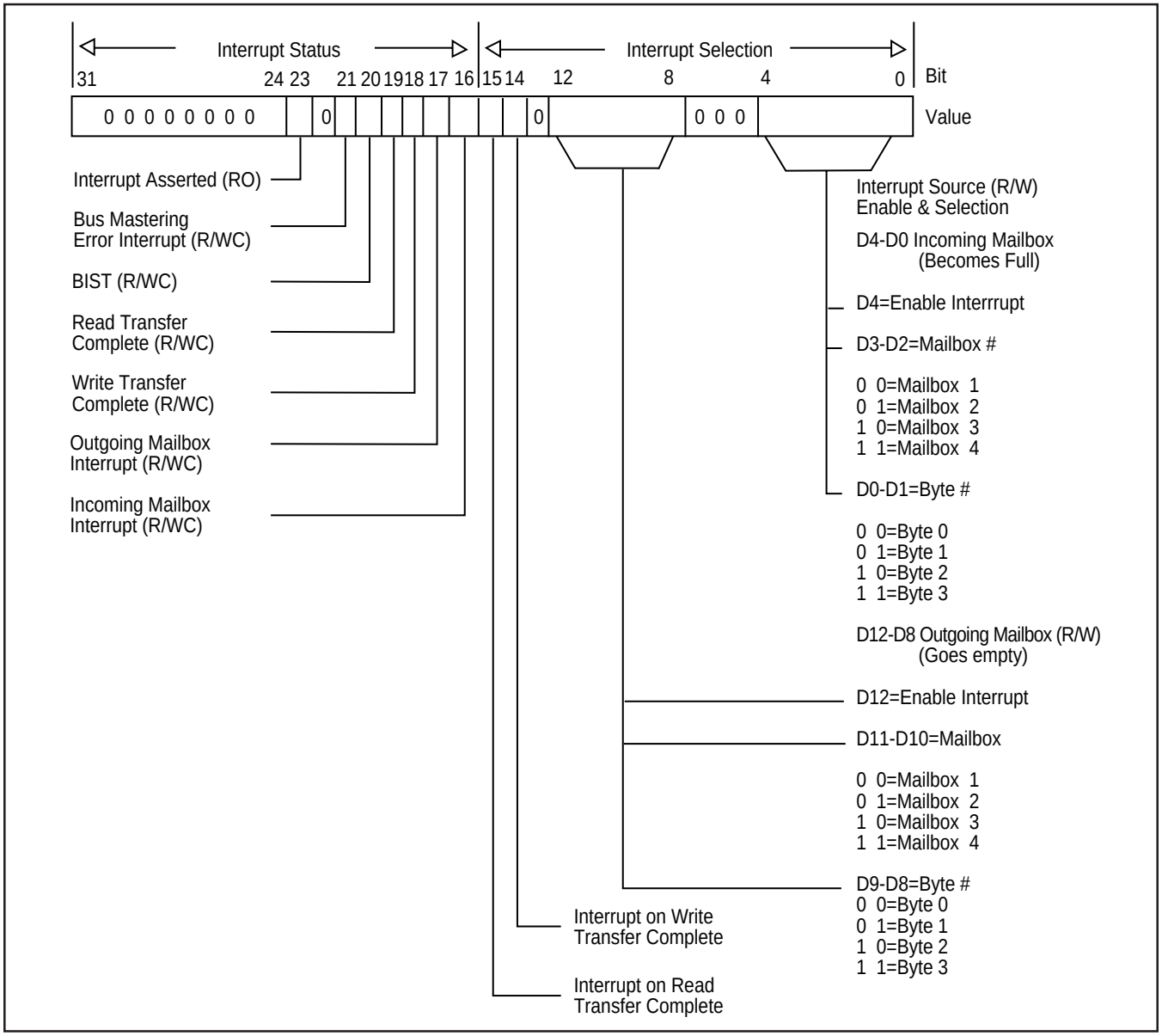


Table 3. Interrupt Control/Status Register

Bit	Description
31:24	Reserved. Always zero.
23	Interrupt asserted. This read-only status bit indicates that one or more interrupt conditions is present. This bit is nothing more than the ORing of the interrupt conditions described by bits, 20, 17 and 16 of this register.
22	Reserved. Always zero.
21	Master/Target Abort. This bit signifies that an interrupt has been generated due to the S5933 encountering a Master or Target abort during an S5933 initiated PCI bus cycle. This bit operates as read or write one clear. Writing a one to this bit causes it to be cleared. Writing a zero to this bit does nothing.
20	BIST. Built-In Self-Test interrupt. This interrupt occurs when a self test is initiated by the PCI interface writing of the BIST configuration register. This bit will stay set until cleared by writing a one to this location. Self test completion codes may be passed to the PCI BIST register by writing to the AGCSTS register.
19	Read Transfer Complete. This bit signifies that an interrupt has been generated due to the completion of a PCI bus master operation involving the transfer of data from the PCI bus to the Add-On. This interrupt will occur when the Master Read Transfer Count register reaches zero. This bit operates as read or write one clear. A write to this bit with the data of one will cause this bit to be reset; a write to this bit with the data of zero will not change the state of this bit.
18	Write Transfer Complete. This bit signifies that an interrupt has been generated due to the completion of a PCI bus master operation involving the transfer of data to the PCI bus from the Add-On. This interrupt will occur when the Master Write Transfer Count register reaches zero. This bit operates as read or write one clear. A write to this bit with the data of one will cause this bit to be reset; a write to this bit with the data of zero will not change the state of this bit.
17	Outgoing Mailbox Interrupt. This bit sets when the mailbox selected by bits 12 through 8 of this register is read by the PCI interface. This bit operates as read or write one clear. A write to this bit with the data as one will cause this bit to be reset; a write to this bit with the data as zero will not change the state of this bit.
16	Incoming Mailbox Interrupt. This bit sets when the mailbox selected by bits 5 through 0 of this register are written by the PCI interface. This bit operates as read or write one clear. A write to this bit with the data of one will cause this bit to be reset; a write to this bit with the data as zero will not change the state of this bit.
15	Interrupt on Read Transfer Complete. This bit enables the occurrence of an interrupt when the read transfer count reaches zero. This bit is read/write.
14	Interrupt on Write Transfer Complete. This bit enables the occurrence of an interrupt when the write transfer count reaches zero. This bit is read/write.
13	Reserved. Always zero.
12	Enable outgoing mailbox interrupt. This bit allows a read by the PCI of the outgoing mailbox register identified by bits 11 through 8 to produce an Add-On interface interrupt. This bit is read/write.
11:10	Outgoing Mailbox Interrupt Select. This field selects which of the four outgoing mailboxes is to be the source for causing an outgoing mailbox interrupt. [00]b selects mailbox 1, [01]b selects mailbox 2, [10]b selects mailbox 3 and [11]b selects mailbox 4. This field is read/write.
9:8	Outgoing Mailbox Byte Interrupt select. This field selects which byte of the mailbox selected by bits 11 and 10 above is to actually cause the interrupt. [00]b selects byte 0, [01]b selects byte 1, [10]b selects byte 2, and [11]b selects byte 3. This field is read/write.
7:5	Reserved. Always zero.

Table 3. Interrupt Control/Status Register (Continued)

Bit	Description
4	Enable incoming mailbox interrupt. This bit allows a write from the PCI bus to the incoming mailbox register identified by bits 3 through 0 to produce an Add-On interface interrupt. This bit is read/write.
3:2	Incoming Mailbox Interrupt Select. This field selects which of the four incoming mailboxes is to be the source for causing an incoming mailbox interrupt. [00]b selects mailbox 1, [01]b selects mailbox 2, [10]b selects mailbox 3 and [11]b selects mailbox 4. This field is read/write.
1:0	Incoming Mailbox Byte Interrupt select. This field selects which byte of the mailbox selected by bits 3 and 2 above is to actually cause the interrupt. [00]b selects byte 0, [01]b selects byte 2, and so on.

ADD-ON GENERAL CONTROL/STATUS REGISTER (AGCSTS)

Register Name: Add-On General Control and Status

Add-On Address Offset: 3Ch

Power-up value: 000000F4h
(PCI initiated bus mastering)
00000034h
(Add-On initiated bus mastering)

Attribute: Read/Write, Read Only, Write Only

Size: 32 bits

This register provides for overall control of the Add-On portion of this device. It is used to provide a method to perform software resets of the mailbox and FIFO flags.

The following Add-On controls are provided:

- Reset PCI to Add-On FIFO flags
- Reset Add-On to PCI FIFO flags
- Reset mailbox empty full status flags
- Write/read external non-volatile memory.

The following status flags are provided to the Add-On:

- Add-On to PCI FIFO FULL
- Add-On to PCI FIFO has four or more empty locations
- Add-On to PCI FIFO EMPTY
- PCI to Add-On FIFO FULL
- PCI to Add-On FIFO has four or more words loaded
- PCI to Add-On FIFO EMPTY

Figure 5. Add-On General Control/Status Register

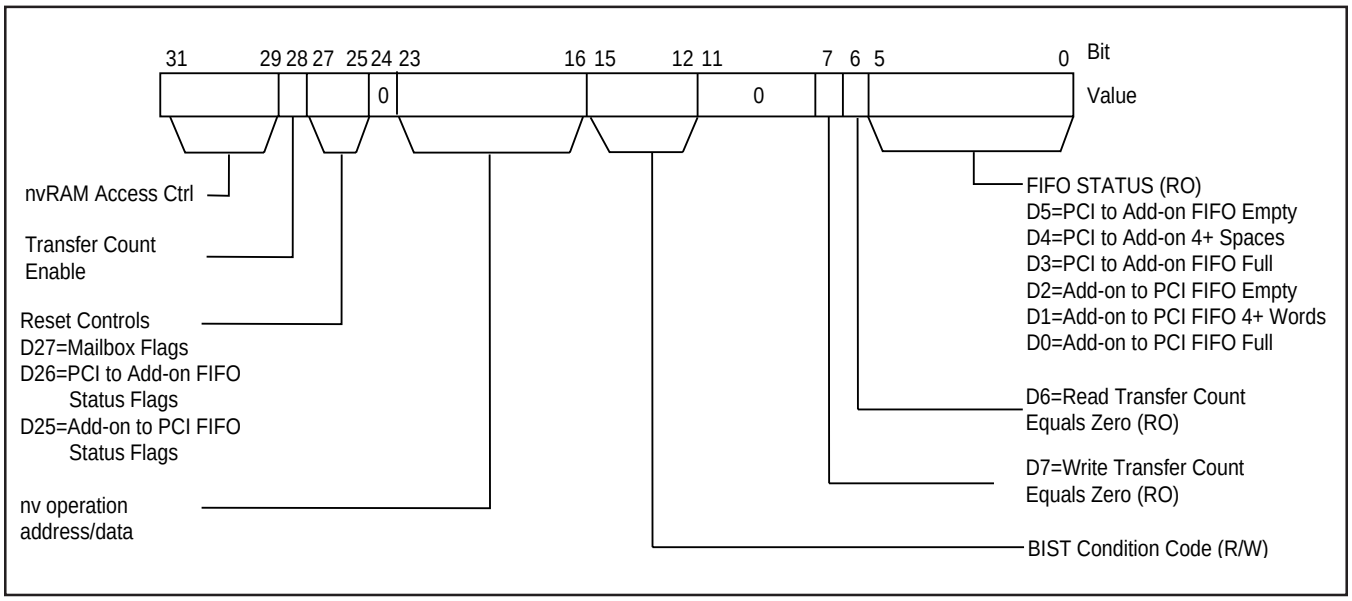


Table 4. Add-On General Control/Status Register

Bit	Description
31:29	nvRAM/EPROM Access Control. This field provides a method for access to the optional, external non-volatile memory. Write operations are achieved by a sequence of byte operations involving these bits and the 8-bit field of bits 23 through 16. The sequence requires that the low-order address, high-order address, and then a data byte be loaded in order. Bit 31 of this field acts as an enable/clock and ready for the access to the external memory. D31 must be written to a 1 before an access can begin, and subsequent accesses must wait for bit D31 to become zero (ready). D31 D30 D29 W/R 0 X X W Inactive 1 0 0 W Load low address byte 1 0 1 W Load high address byte 1 1 0 W Begin write 1 1 1 W Begin read 0 X X R Ready 1 X X R Busy Cautionary note: The non-volatile memory interface is also available for access by the PCI bus interface. Accesses by both the Add-On and PCI bus to the nv memory are not directly supported by this component. Software must be designed to prevent the simultaneous access of nv memory to prevent data corruption within the memory and provide for accurate data retrieval.
28	Transfer Count Enable. When set, transfer counts are used for Add-On initiated bus master transfers. When clear, transfer counts are ignored.
27	Mailbox Flag Reset. Writing a 1 to this bit causes all mailbox status flags to become reset (EMPTY). It is not necessary to write this bit as 0 because it is used internally to produce a reset pulse. Since reading of this bit will always produce zeros, this bit is write only.
26	PCI to Add-On FIFO Status Reset. Writing a 1 to this bit causes the Inbound (Bus master reads) FIFO empty flag to set indicating empty and the FIFO FULL flag to reset and the FIFO Four Plus spaces flag to set. It is not necessary to write this bit as 0 because it is used internally to produce a reset pulse. Since reading of this bit would always produce zeros, this bit is write only.
25	Add-On to PCI FIFO Status Reset. Writing a one to this bit causes the Outbound (Bus master writes) FIFO empty flag to set indicating empty and the FIFO FULL flag to reset and the FIFO Four Plus words available flag to reset. It is not necessary to write this bit as zero because it is used internally to produce a reset pulse. Since reading of this bit would always produce zeros, this bit is write only.
24	Reserved. Always zero.
23:16	Non-volatile memory address/data port. This 8-bit field is used in conjunction with bit 31, 30 and 29 of this register to access the external non-volatile memory. The contents written are either low address, high address, or data as defined by bits 30 and 29. This register will contain the external non-volatile memory data when the proper read sequence for bits 31 through 29 is performed.
15:12	BIST condition code. This field is directly connected to the PCI configuration self test register. Bit 15 through 12 maps with the BIST register bits 3 through 0, respectively.
11:8	Reserved. Always zero.

Table 4. Add-On General Control/Status Register (Continued)

Bit	Description
7	Add-On to PCI Transfer Count Equal Zero (RO). This bit as a one signifies that the write transfer count is all zeros. Only when Add-On initiated bus mastering is enabled.
6	PCI to Add-On Transfer Count Equals Zero (RO). This bit as a one signifies that the read transfer count is all zeros. Only when Add-On initiated bus mastering is enabled.
5	PCI to Add-On FIFO Empty. This bit is a 1 when the PCI to Add-On FIFO is empty.
4	PCI to Add-On FIFO 4+ spaces. This bit is a 1 when there are four or more open spaces in the PCI to Add-On FIFO.
3	PCI to Add-On FIFO Full. This bit is a 1 when the PCI to Add-On FIFO is full.
2	Add-On to PCI FIFO Empty. This bit is a 1 when the Add-On to PCI FIFO is empty.
1	Add-On PCI FIFO 4+ words. This bit is a 1 when there are four or more full locations in the Add-On to PCI FIFO.
0	Add-On to PCI FIFO Full. This bit is a 1 when the Add-On to PCI FIFO is full.

ADD-ON CONTROLLED BUS MASTER WRITE TRANSFER COUNT REGISTER (MWTC)

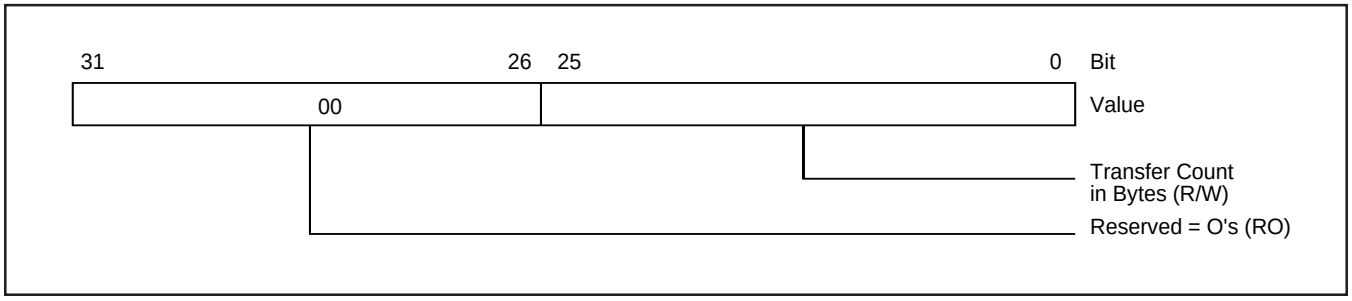
Register Name: Master Write Transfer Count
Add-On Address Offset: 58h
Power-up value: 00000000h
Attribute: Read/Write
Size: 32 bits

This register is only accessible when Add-On initiated bus mastering is enabled.

The master write transfer count register is used to convey to the S5933 controller the actual number of bytes that are to be transferred. The value in this register is decremented with each bus master PCI write operation until the transfer count reaches zero.

Upon reaching zero, the transfer operation ceases and an interrupt may be optionally generated to either the PCI or Add-On bus interface. Transfers which are not whole multiples of DWORDs in size result in a partial word ending cycle. This partial word ending cycle is possible since all bus master transfers for this controller are required to begin on a DWORD boundary.

Figure 6. Add-On Controlled Bus Master Write Transfer Count Register



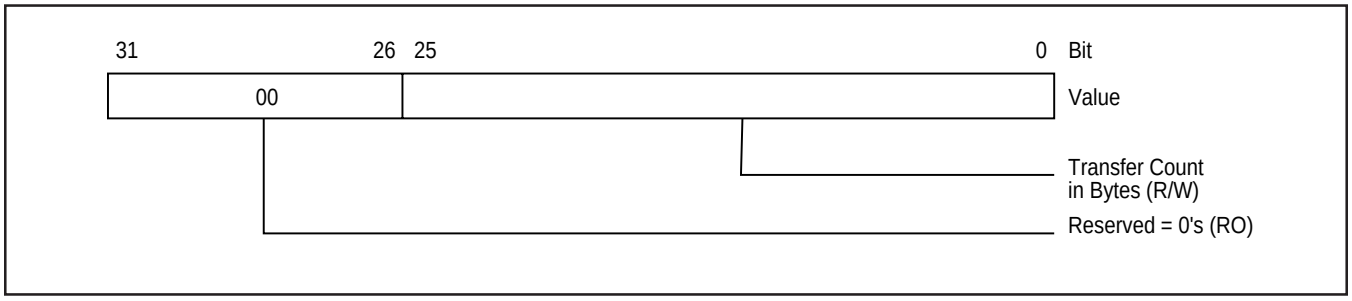
ADD-ON CONTROLLED BUS MASTER READ TRANSFER COUNT REGISTER (MRTC)

Register Name: Master Read Transfer Count
Add-On Address Offset: 5Ch
Power-up value: 00000000h
Attribute: Read/Write
Size: 32 bits

This register is only accessible when Add-On initiated bus mastering is enabled.

The master read transfer count register is used to convey to the PCI controller the actual number of bytes that are to be transferred. The value in this register is decremented with each bus master PCI read operation until the transfer count reaches zero. Upon reaching zero, the transfer operation ceases and an interrupt may be optionally generated to either the PCI or Add-On bus interface. Transfers which are not whole multiples of DWORDs in size result in a partial word ending cycle. This partial word ending cycle is possible since all bus master transfers for this controller are required to begin on a DWORD boundary.

Figure 7. Add-On Controlled Bus Master Read Transfer Count Register



INITIALIZATION

All PCI bus agents and bridges are required to implement PCI Configuration Registers. When multiple PCI devices are present, these registers must be unique to each device in the system. The specified PCI procedure for uniquely selecting a device's configuration space involves a dedicated signal, called IDSEL, connected to each motherboard PCI bus device and PCI slot.

The host executes configuration cycles after reset to each device on the PCI bus. The configuration registers provide information on PCI agent operation and memory or I/O space requirements. These allow the PCI BIOS to enable the device and locate it within system memory or I/O space.

After a PCI reset, the S5933 can be configured for a specific application by downloading device setup information from an external non-volatile memory into the device Configuration Registers. The S5933 can also be used in a default configuration, with no external boot device.

When using a non-volatile boot memory to customize operation, 64 bytes are required for S5933 setup information. The rest of the boot device may be used to implement an Expansion BIOS, if desired. Some of the setup information is used to initialize the S5933 PCI Configuration Registers, other information is not downloaded into registers, but is used to define S5933 operation (FIFO interface, Pass-Thru operation, etc.).

PCI RESET

Immediately following the assertion of the PCI RST# signal, the Add-On reset output SYSRST# is asserted. Immediately following the deassertion of RST#, SYSRST# is deasserted. The Add-On reset output may be used to initialize state machines, reset Add-On microprocessors, or reset other Add-On logic devices.

All S5933 Operation Registers and Configuration Registers are initialized to their default states at reset. The default values for the Configuration Registers may be overwritten with the contents of an external nv boot memory during device initialization, allowing a custom device configuration. Configuration accesses by the host CPU to the S5933 produce PCI bus wait states until one of the following events occurs:

- The S5933 identifies that there is no valid boot memory (and default Configuration Register values are used).
- The S5933 finishes downloading all configuration information from a valid boot memory.

LOADING FROM BYTE-WIDE NV MEMORIES

The SNV input on the S5933 indicates what type of external boot-load device is present (if any). If SNV is tied low, a byte-wide nv memory is assumed. In this case, immediately after the PCI bus reset is deasserted, the address 0040h is presented on the nv memory interface address bus EA[15:0]. Eight PCI clocks later (240 ns at 33 MHz), data is read from the nv memory data bus EQ[7:0] and address 0041h is presented. After an additional eight PCI clocks, data is again read from EQ7:0. If both accesses read are all ones (FFh), it implies an illegal Vendor ID value, and the external nv memory is not valid or not present. In this situation, the AMCC default configuration values are used.

If either of the accesses to address 0040h and 0041h contain zeros (not FFh), the next accesses are to locations 0050h, 0051h, 0052h, and 0053h. At these locations, the data must be C0h (or C1h or C2h), FFh, E8h, and 10h, respectively, for the external nv memory to be valid. Once a valid external nv memory has been recognized, it is read, sequentially, from location 0040h to 007Fh. The appropriate data is loaded into the PCI Configuration Registers as described in Chapter 4. Some of the boot device data is not downloaded into Configuration Registers, but is used to enable features and configure S5933 operation. Upon completion of this procedure, the boot-load sequence terminates and PCI configuration accesses to the S5933 are acknowledged with the PCI Target Ready (TRDY#) output.

Table 1 lists the required nv memory contents for a valid configuration nv memory device.

Table 1. Valid External Boot Memory Contents

Address	Data	Notes
0040h-41h	not FFFFh	This is the location that the S5933 PCI Controller will load a customized vendor ID. (FFFFh is an illegal vendor ID.)
0050h	C2h, C1h or C0h	This is the least significant byte of the region which initializes the base address register #0 of the S5933 configuration register (Section 3.11). A value of C1h assigns the 16 DWORD locations of the PCI operation registers into I/O space, a value of C0h defines memory space, a value of C2h defines memory space below 1 Mbyte.
0051	FFh	Required.
0052h	E8h	Required.
0053h	10h	Required.

LOADING FROM SERIAL NV MEMORIES

SNV tied high indicates that a serial nv memory (or no external device) is present. When serial nv memories are used, data transfer is performed through a two-wire, bidirectional data transfer protocol as defined by commercial serial EEPROM/Flash offerings. These devices have the advantages of low pin counts, small package size, and economical price.

A serial nv memory is considered valid if the first serial accesses contain the correct per-byte acknowledgments (see Figure 3). If the serial per-byte acknowledgment is not observed, the S5933 determines that no external serial nv memory is present and the AMCC default Configuration Register values are used.

Two pins are used to transfer data between the S5933 PCI controller and the external serial memory: a serial clock pin, SCL, and a serial data pin, SDA. The serial clock pin is an output from the S5933, and the serial data pin is bidirectional. The serial clock is derived by dividing the PCI bus clock by 512. This means that the frequency of the serial clock is approximately 65 kHz for a 33-MHz PCI bus clock.

Communications with the serial memory involve several clock transitions. A start event signals the beginning of a transaction and is immediately followed by an address transfer. Each address/data transfer consists of 8 bits of information followed by a 1-bit acknowledgment. When the exchange is complete, a stop event is issued. Figure 1 shows the unique relationship defining both a start and stop event. Figure 2 shows the required timing for address/data with respect to the serial clock.

For random accesses, the sequence involves one clock to define the start of the sequence, eight clocks to send the slave address and read/write command, followed by a one-clock acknowledge, and so on. Figure 3 shows the sequence for a random write access requiring 29 serial clock transitions. At the clock speed for the S5933, this corresponds to one byte of data transferred approximately every 0.5 milliseconds. Read accesses may be either random or sequential. Random read access requires a dummy write to load the word address and require 39 serial clock transitions. Figure 4 shows the sequence for a random byte read.

To initialize the S5933 controller's PCI Configuration Registers, the smallest serial device necessary is a 128 x 8 organization. Although the S5933 controller only requires 64 bytes, these bytes must begin at a 64-byte address offset (0040h through 007Fh). This offset constraint permits the configuration image to be shared with a memory containing expansion BIOS code and the necessary preamble to identify an expansion BIOS. The largest serial device which may be used is 2 Kbytes.

Figure 1. Serial Interface Definition of Start and Stop

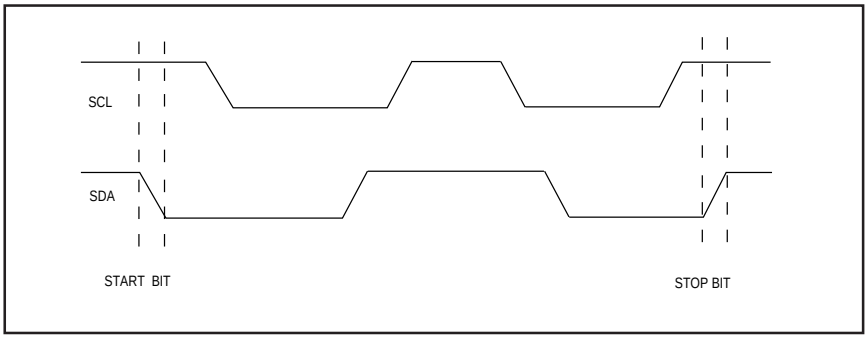


Figure 2. Serial Interface Clock/Data Relationship

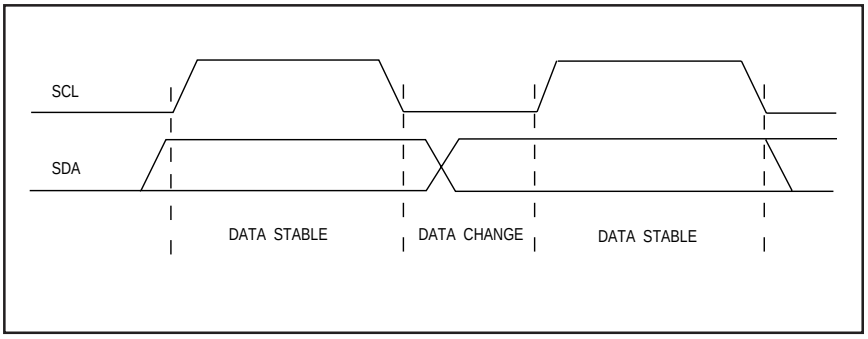


Figure 3. Serial Interface Byte Access — Write

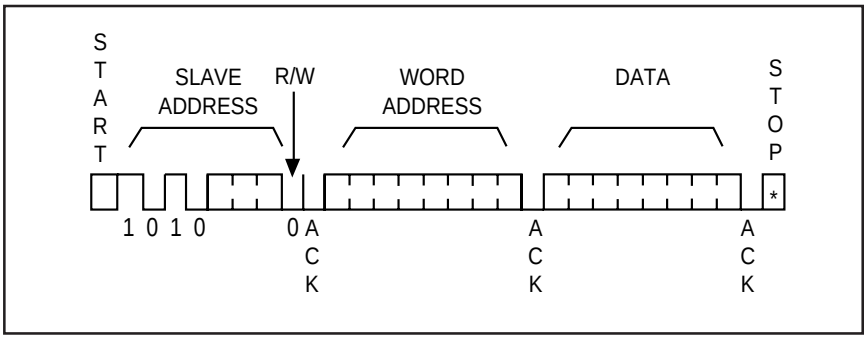
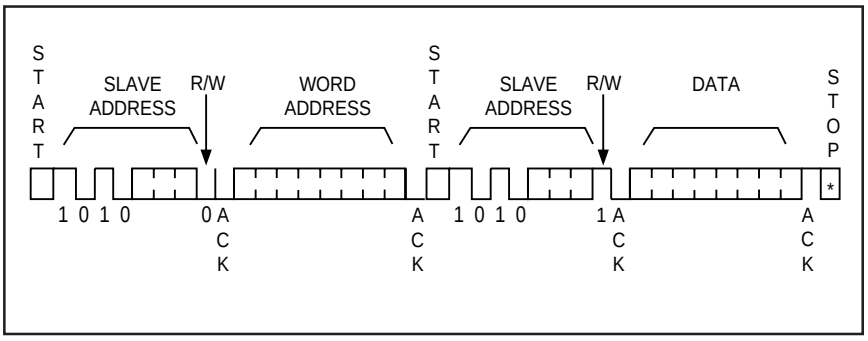


Figure 4. Serial Interface Byte Access — Read



PCI BUS CONFIGURATION CYCLES

Cycles beginning with the assertion IDSEL and FRAME# along with the two configuration command states for C/BE[3:0] (configuration read or write) access an individual device's configuration space. During the address phase of the configuration cycle just described, the values of AD0 and AD1 identify if the access is a Type 0 configuration cycle or a Type 1 configuration cycle. Type 0 cycles have AD0 and AD1 equal to 0 and are used to access PCI bus agents. Type 1 configuration cycles are intended only for bridge devices and have AD0 as a 1 with AD1 as a 0 during the address phase.

The S5933 PCI device is a bus agent (not a bridge) and responds only to a Type 0 configuration accesses. Figure 5 depicts the state of the AD bus during the address phase of a Type 0 configuration access. The S5933 controller does not support the multiple function numbers field (AD[10:8]) and only responds to the all-zero function number value.

The configuration registers for the S5933 PCI controller can only be accessed under the following conditions:

- IDSEL high (PCI slot unique signal which identifies access to configuration registers) along with FRAME# low.
- Address bits A0 and A1 are 0 (Identifies a Type 0 configuration access).
- Address bits A31-A11 are ignored.
- Address bits A8, A9, and A10 are 0 (Function number field of zero supported).
- Command bits, C/BE[3:0]# must identify a configuration cycle command (101X).

Figure 6 describes the signal timing relationships for configuration read cycles. Figure 7 describes configuration write cycles.

Figure 5. PCI AD Bus Definition During a Type 0 Configuration Access

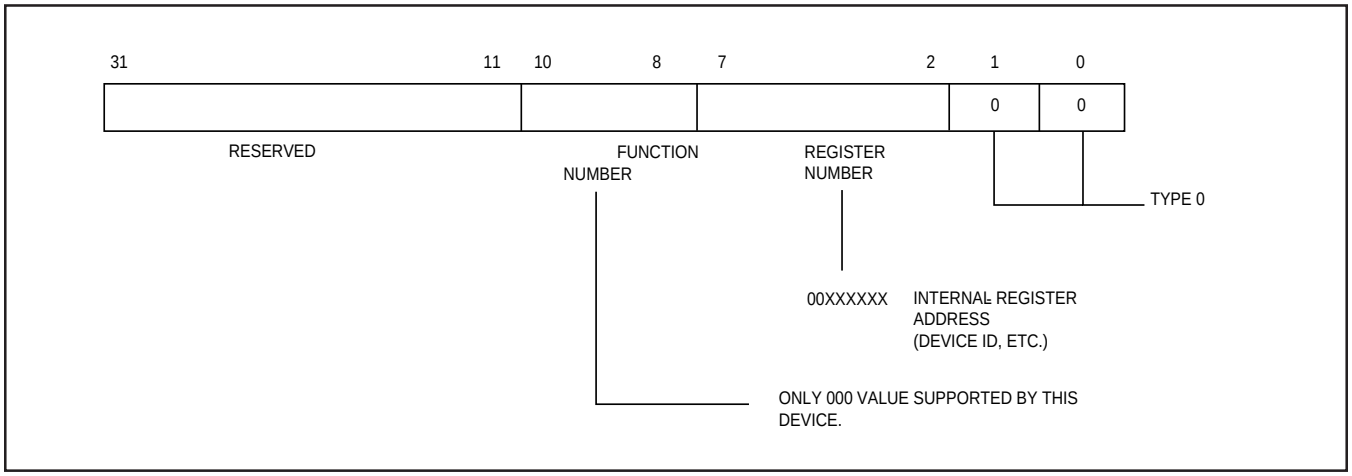


Figure 6. Type 0 Configuration Read Cycles

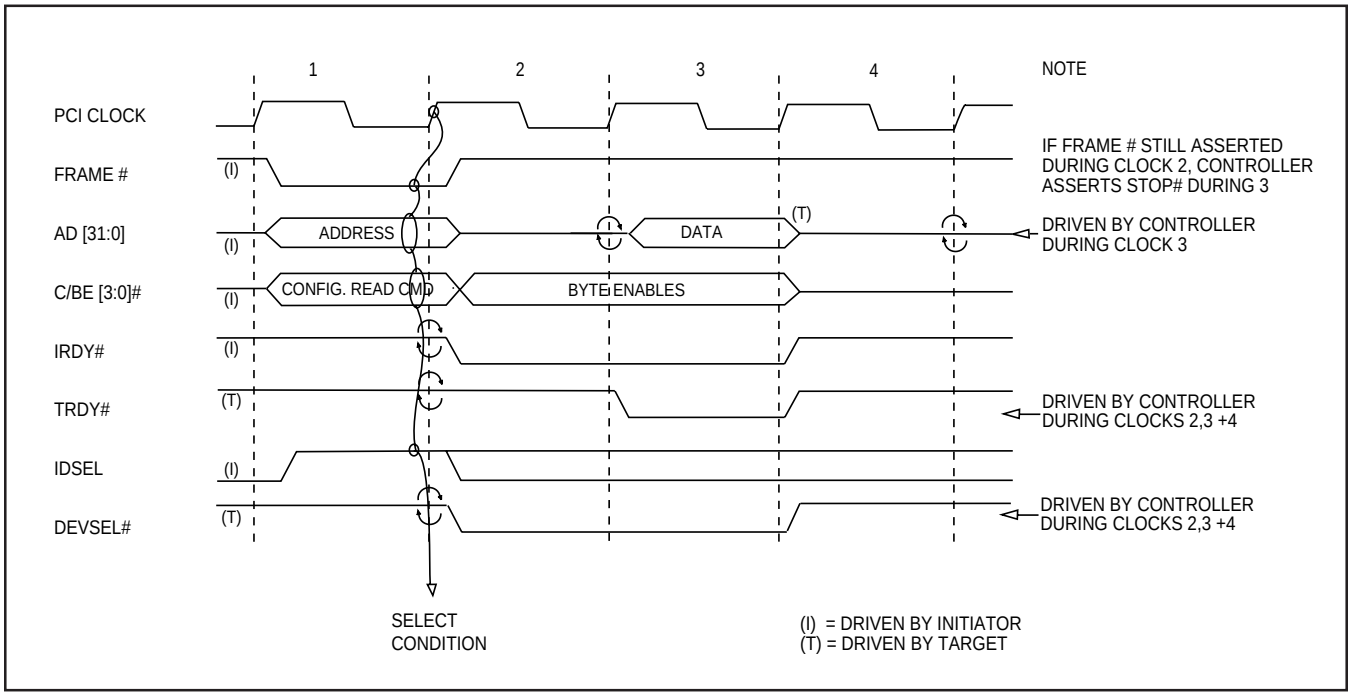
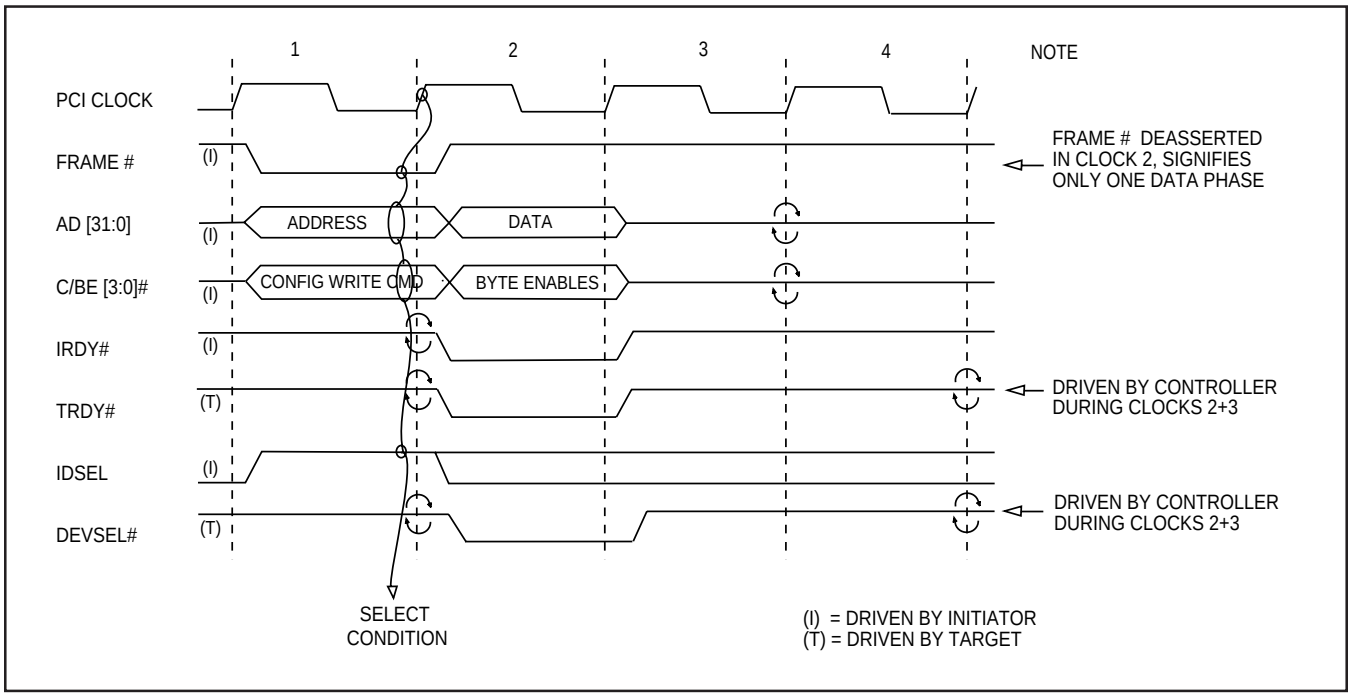


Figure 7. Type 0 Configuration Write Cycles



EXPANSION BIOS ROMS

This section provides an example of a typical PC-compatible expansion BIOS ROM. Address offsets 0040h through 007Fh represent the portion of the external nv memory used to boot-load the S5933 controller. Whether the expansion ROM is intended to be executable code is determined by the contents of the first three locations (starting at offset 0h) and a byte checksum over the defined length. The defined length is specified in the byte at address offset 0002h. Table 2 lists each field location by its address offset, its length, its value, and description.

Table 2. PC Compatible Expansion ROM

Byte Offset	Byte Length	Binary Value	Description	Example
0h	1	55h	BIOS ROM signature byte 1	55h
1h	1	AAh	BIOS ROM signature byte 2	AAh
2h	1	var.	Length in multiples of 512 bytes	01h
3h	4	var.	Entry point for INIT function.	
7h-17h	17h	var.	Reserved (application unique data)	
18h-19h	2	var.	Pointer to PCI Data Structure (see Table 3)	
20h-3Fh	32h	var	user-defined	
The following represents the boot-load image for the S5933 controller's PCI configuration register :				
40h	2	[your vendor ID]		10e8h
42h	2	[your device ID]		4750h
44h	1	not used		
45h	1	[Bus Master Config.]		80h
46h	2	not used		
48h	1	[your revision ID]		00h
49h	3	[your class code]		FF0000h
4Ch	1	not used		
4Dh	1	[your latency timer #]		00h
4Eh	1	[your header type]		00h
4Fh	1	[self-test if desired]		80h or 00h
50h	1	C0h, C1h or C2h		C0h, C1h or C2h
51h	1	FFh		FFh
52h	1	E8h		E8h
53h	1	10h		10h
54h	4	[base addr. #1]		xxxxxxxxh
58h	4	[base addr. #2]		xxxxxxxxh
5Ch	4	[base addr. #3]		xxxxxxxxh
60h	4	[base addr. #4]		xxxxxxxxh
64h	4	[base addr. #5]		xxxxxxxxh

Table 2. PC Compatible Expansion ROM (Continued)

Byte Offset	Byte Length	Binary Value	Description	Example
68h	8	not used		
70h	4	[Expansion ROM base addr.]	(example shows 32K bytes)	FFFF8001h
74h	8	not used		
7Ch	1	[Interrupt line]		0Ch
7Dh	1	[Interrupt pin]		01h
7Eh	1	[Min-Grant]		00h
7Fh	1	[Max_lat]		00h
80h — (1FFh), or (2FFh), or (3FFh), etc.	application specific Byte checksum, location dependent on value for length field at offset 0002h.			

A 16-bit pointer at location 18h of the PC expansion ROM identifies the start offset of the PCI data structure. The PCI data structure is shown in Table 3 and contains various vendor, product, and program evolutions. If a valid external nv memory is identified by the S5933, the PCI data structure is used to configure the S5933. The PCI data structure is not necessary for this device to operate. If no external nv memory is implemented, the S5933 boots with the default configuration values.

Note: If a serial BIOS ROM is used, the access time for large serial devices should be considered, since it may cause a lengthy system delay during initialization. For example, a 2-Kbyte serial device takes about 1 second to be read. Many systems, even when BIOS ROMs are ultimately shadowed into system RAM, may read this memory space twice (once to validate its size and checksum, and once to move it into RAM). Execution directly from a serial BIOS ROM, although possible, may be unacceptably slow.

Table 3. PCI Data Structure

Byte Offset	Byte Length	Binary Value	Description
0h	4	‘PCIR’	Signature, the ASCII string ‘PCIR’ where ‘P’ is at offset 0, ‘C’ at offset 1, and so on.
4h	2	var.	Vendor Identification
6h	2	var.	Device Identification
8h	2	var.	Pointer to Vital Product Data
Ah	2	var.	PCI Data Structure Length (starts with signature field)
Ch	1	var.	PCI Data Structure Revision (=0 for this definition)
Dh	3	var.	Class Code
10h	2	var.	Image Length
12h	2	var.	Revision Level
14h	1	var.	Code Type
15h	1	var.	Indicator (bit D7=1 signifies “last image”)
16h	2	0000h	Reserved

PCI BUS INTERFACE

This section describes the various events which occur on the S5933 PCI bus interface. Since the S5933 controller functions as both a target (slave) and an initiator (master), signal timing detail is given for both situations this Section presents the signal relationships involved in performing basic read or write transfers on the PCI bus and also describes the different ways these cycles may complete.

PCI BUS TRANSACTIONS

Because the PCI bus has multiplexed address/data pins, AD[31:0], each PCI bus transaction consists of two phases: Address and Data. An address phase is defined by the clock period when the signal FRAME# transitions from inactive (high) to active (low). During the address phase, a bus command is also driven by the initiator on signal pins C/BE[3:0]#. If the command indicates a PCI read, the clock cycle following the address phase is used to perform a "bus turn-around" cycle. A turn-around cycle is a clock period in which the AD bus is not driven by the initiator or the target device. This is used to avoid PCI bus contention. For a write command, a turn-around cycle is not needed, and the bus goes directly from the address phase to the data phase.

All PCI bus transactions consist of an address phase (described above), followed by one or more data phases. The address phase is only one PCI clock long and the bus cycle information (address and command) is latched internally by the S5933. The number of data phases depends on how many data transfers are desired or are possible with a given initiator-target pair. A data phase consists of at least one PCI clock. FRAME# is deasserted to indicate that the final data phase of a PCI cycle is occurring. Wait states may be added to any data phase (each wait state is one PCI clock).

The PCI bus command presented on the C/BE[3:0]# pins during the address phase can represent 16 possible states. Table 1 lists the PCI commands and identifies those which are supported by the S5933 controller as a target and those which may be produced by the S5933 controller as an initiator. A "Yes" in the "Supported As Target" column in Table 1 indicates the S5933 controller asserts the signal DEVSEL# when that command is issued along with the appropriate PCI address. Two commands are supported by the S5933 controller as an initiator: Memory Read and Memory Write.

The completion or termination of a PCI cycle can be signaled in several ways. In most cases, the completion of the final data phase is indicated by the assertion of ready signals from both the target (TRDY#) and initiator (IRDY#) while FRAME# is inactive. In some cases, the target is not be able to continue or support a burst transfer and asserts the STOP# signal. This is referred to as a target disconnect. There are also cases where an addressed device does not exist, and the signal DEVSEL# never becomes active. When no DEVSEL# is asserted in response to a PCI cycle, the initiator is responsible for ending the cycle. This is referred to as a master abort. The bus is returned to the idle phase when both FRAME# and IRDY# are deasserted.

Table 1. Supported PCI Bus Commands

C/BE[3:0]#	Command Type	Supported As Target	Supported As Initiator
0000	Interrupt Acknowledge	No	No
0001	Special Cycle	No	No
0010	I/O Read	Yes	No
0011	I/O Write	Yes	No
0100	Reserved	No	No
0101	Reserved	No	No
0110	Memory Read	Yes	Yes
0111	Memory Write	Yes	Yes
1000	Reserved	No	No
1001	Reserved	No	No
1010	Configuration Read	Yes	No
1011	Configuration Write	Yes	No
1100	Memory Read Multiple	Yes [1]	No [3]
1101	Reserved	No	No
1110	Memory Read Line	Yes [1]	No
1111	Memory Write & Invalidate	Yes [2]	No

1. Memory Read Multiple and Read Line are treated as Memory Reads.
2. Memory Write & Invalidate commands are treated as Memory Writes.
3. Must be enabled by bit 15 MCSR.

PCI Burst Transfers

The PCI bus, by default, expects burst transfers to be executed. To successfully perform a burst transfer, both the initiator and target must order their burst address sequence in an identical fashion. There are two different ordering schemes: linear address incrementing and 80486 cache line fill sequencing. The exact ordering scheme for a bus transaction is defined by the state of the two least significant AD lines during the address phase. The decoding for these lines is shown below:

AD[1:0]	Burst Order
0 0	Linear sequence
0 1	Reserved
1 0	Cacheline Wrap Mode
1 1	Reserved

The S5933 supports both the linear and the cache line burst ordering. When the S5933 controller is an initiator, it always employs a linear ordering.

Some accesses to the S5933 controller (as a target) can not be burst transfers. For example, the S5933 does not allow burst transfers when accesses are made to the configuration or operation registers (including the FIFO as a target). Attempts to perform burst transfers to these regions cause STOP# to be asserted during the first data phase. The S5933 completes the initial data phase successfully, but asserting STOP# indicates that the next access needs to be a completely new cycle. Accesses to memory or I/O regions defined by the Base Address Registers 1-4 may be bursts, if desired.

PCI Read Transfers

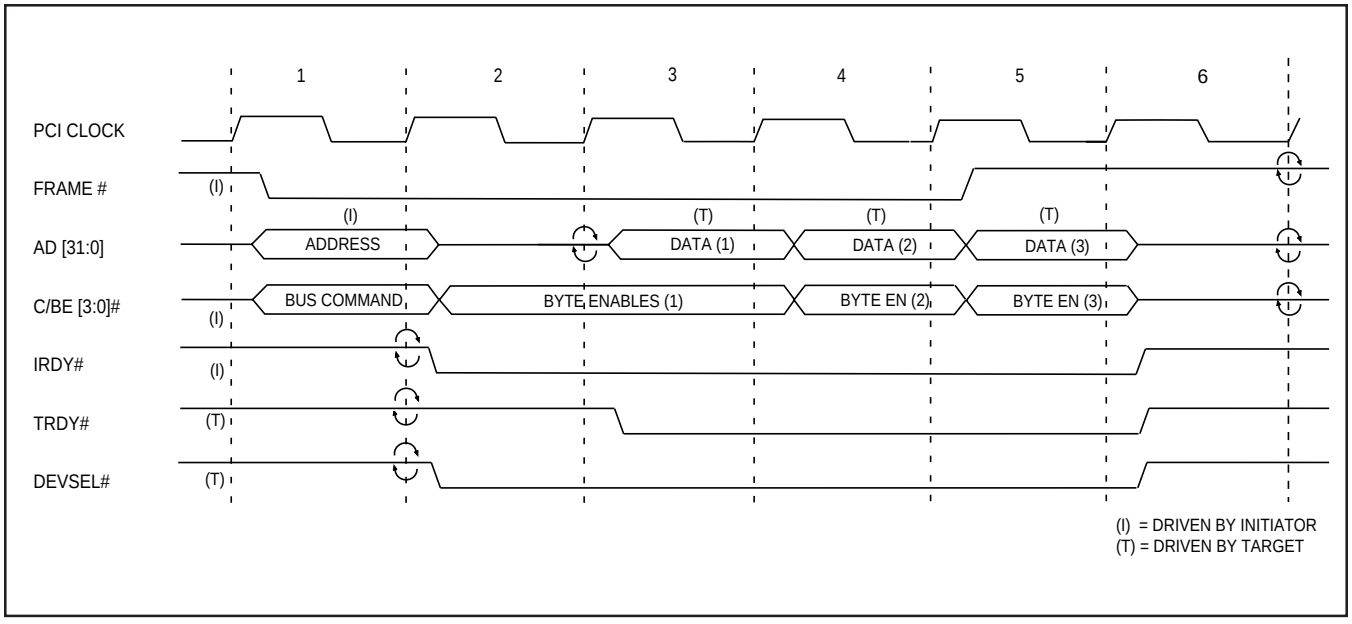
The S5933 responds to PCI bus memory or I/O read transfers when it is selected (target). As a PCI bus initiator, the S5933 controller may also produce PCI bus memory read operations.

Figure 1 depicts the fastest burst read transfer possible for the PCI bus. The timings shown in Figure 1 are representative of the S5933 as a PCI initiator with a fast, zero-wait-state memory target. The signals driven by the S5933 during the transfer are FRAME#, C/BE[3:0]#, and IRDY#. The signals driven by the target are DEVSEL# and TRDY#. AD[31:0] are driven by both the target and initiator during read transactions (only one during any given clock). Clock period 2 is a required bus turn-around clock which ensures bus contention between the initiator and target does not occur.

Targets drive DEVSEL# and TRDY# after the end of the address phase (boundary of clock periods 1 and 2 of Figure 1). TRDY# is not driven until the target can provide valid data for the PCI read. When the S5933 becomes the PCI initiator, it attempts to perform sustained zero-wait state burst reads until one of the following occurs:

- The memory target aborts the transfer
- PCI bus grant (GNT#) is removed
- The PCI to Add-On FIFO becomes full
- A higher priority (Add-On to PCI) S5933 transfer is pending (if programmed for priority)
- The read transfer byte count reaches zero
- Bus mastering is disabled from the Add-On interface

Figure 1. Zero Wait State Burst Read PCI Bus Transfer (S5933 as Initiator)



Read accesses from the S5933 operation registers (S5933 as a target) are shown in Figure 2. The S5933 conditionally asserts STOP# in clock period 3 if the initiator keeps FRAME# asserted during clock period 2 with IRDY# asserted (indicating a burst is being attempted). Wait states may be added by the initiator by not asserting the signal IRDY# during clock 3 and beyond. If FRAME# remains asserted, but IRDY# is not asserted, the initiator is just adding wait states, not necessarily attempting a burst.

There is only one condition where accesses to S5933 operation registers do not return TRDY# but do assert STOP#. This is called a target-initiated termination or target disconnect and occurs when a read attempt is made to an empty S5933 FIFO. The assertion of STOP# without the assertion of TRDY# indicates that the initiator should retry the operation later.

When burst read transfers are attempted to the S5933 operation registers, STOP# is asserted during the first data transfer to indicate to the initiator that no further transfers (data phases) are possible. This is a target-initiated termination where the target disconnects after the first data transfer. Figure 3 shows the signal relationships during a burst read attempt to the S5933 operation registers.

Figure 2. Single Data Phase PCI Bus Read of S5933 Registers (S5933 as Target)

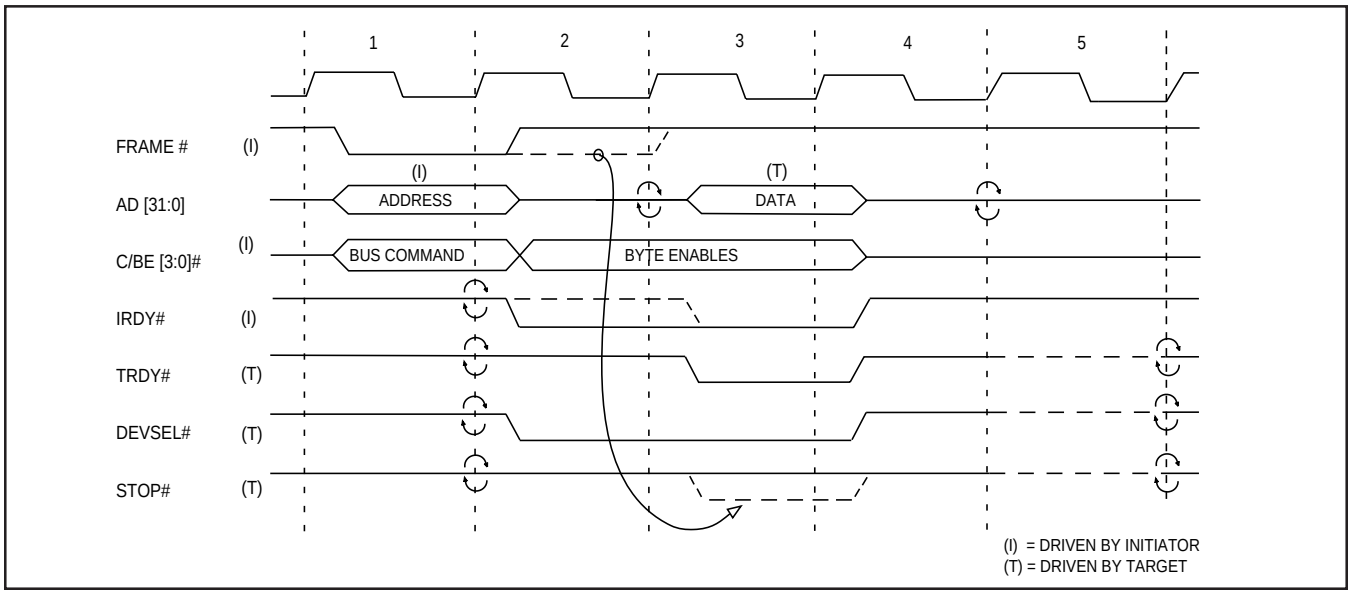
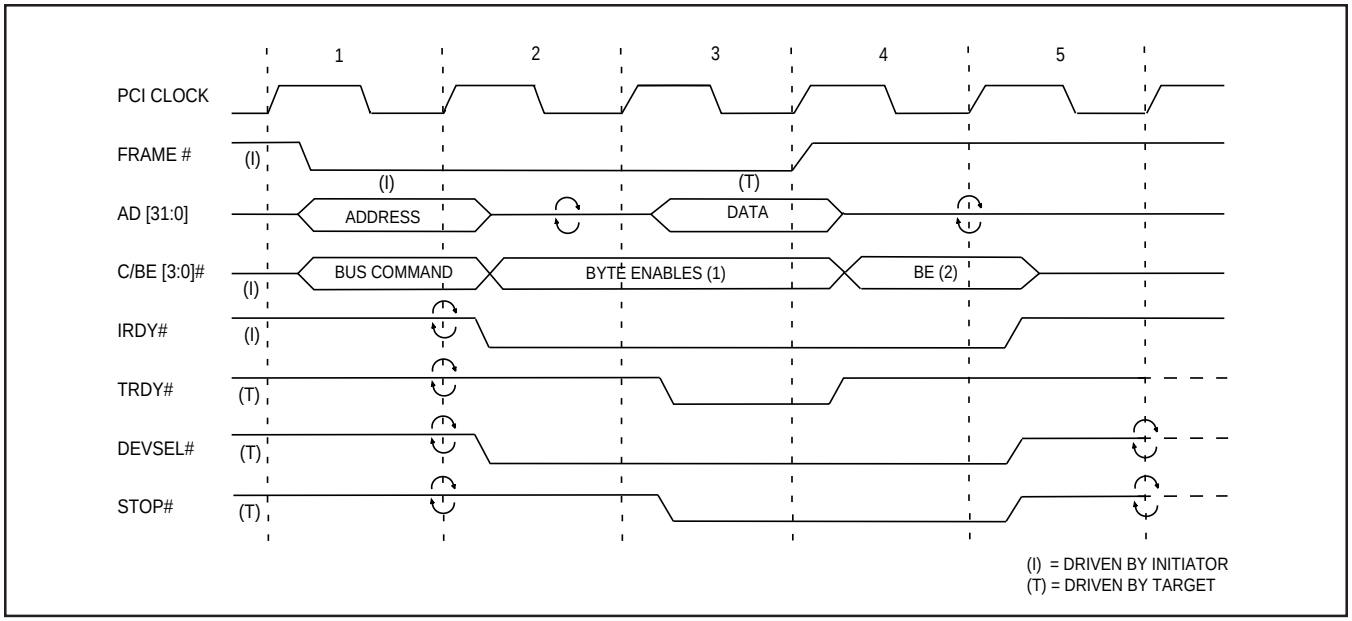


Figure 3. Burst PCI Bus Read Attempt to S5933 Registers (S5933 as Target)



PCI Write Transfers

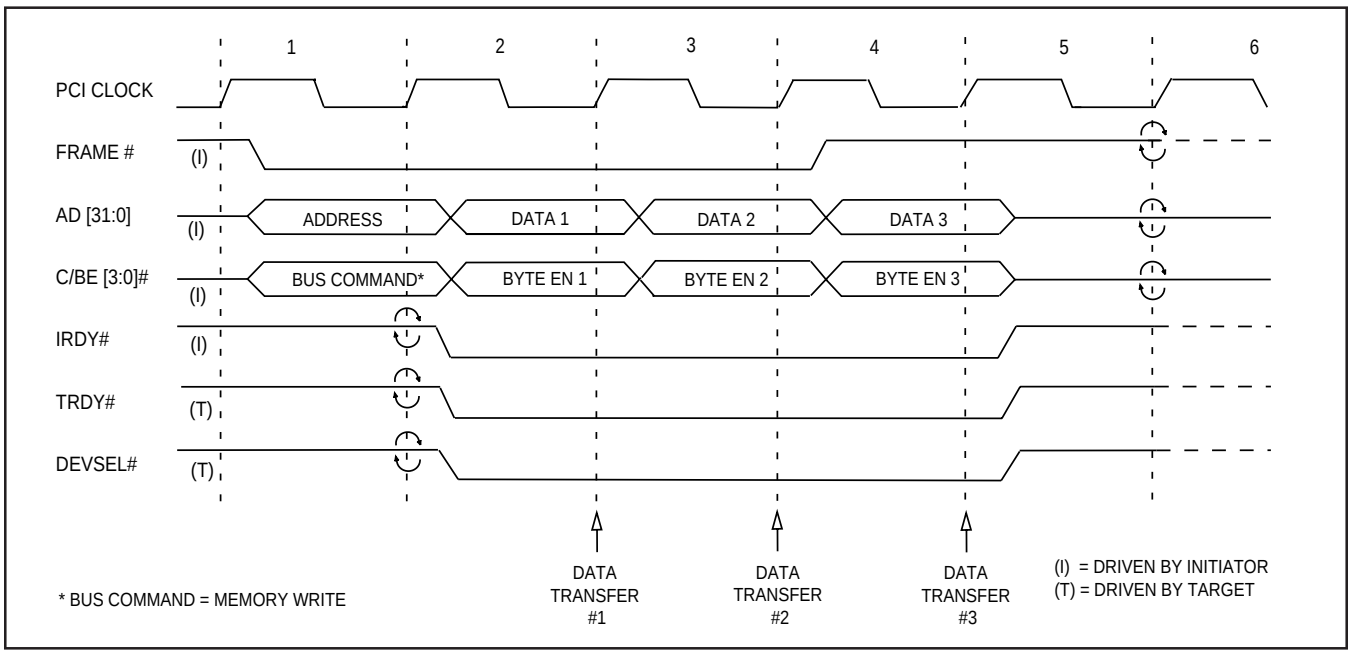
Write transfers on the PCI bus are one clock period shorter than read transfers. This is because the AD[31:0] bus does not require a turn-around cycle between the address and data phases. When the S5933 is accessed (target), it responds to a PCI bus memory or I/O transfers. As a PCI initiator, the S5933 controller can also execute PCI memory write operations.

The timing diagram in Figure 4 represents an S5933 initiator PCI write operation transferring to a fast, zero-wait-state memory target. The signals driven by the S5933 during the transfer are FRAME#, AD[31:0], C/BE[3:0]#, and IRDY#. The signals driven by the target are DEVSEL# and TRDY#. As with PCI reads, targets assert DEVSEL# and TRDY# after the clock defining the end of the address phase (boundary of clock periods 1 and 2 of Figure 4). TRDY# is not driven until the target has accepted the data for the PCI write. When the S5933 becomes the PCI initiator, it attempts sustained zero-wait state burst writes until one of the following occurs:

- The memory target aborts the transfer
- PCI bus grant (GNT# is removed)
- The Add-On to PCI FIFO becomes empty
- A higher priority (PCI to Add-On) S5933 transfer is pending (if programmed for priority)
- The write transfer byte count reaches zero
- Bus mastering is disabled from the Add-On interface

Write accesses to the S5933 operation registers (S5933 as a target) are shown in Figure 5. Here, the S5933 asserts the signal STOP# in clock period 3. STOP# is asserted because the S5933 supports fast, zero-wait-state write cycles but does not support burst writes to operation registers. Wait states may be added by the initiator by not asserting the signal IRDY# during clock 2 and beyond. There is only one condition where writes to S5933 operation registers do not return TRDY# (but do assert STOP#). This is called a target-initiated termination or target disconnect and occurs when a write attempt is made to a full S5933 FIFO. As with the read transfers, the assertion of STOP# without the assertion of TRDY# indicates the initiator should retry the operation later.

Figure 4. Zero Wait State Burst Write PCI Bus Transfer (S5933 as Initiator)



Master-Initiated Termination

Occasionally, a PCI transfer must be terminated by the initiator. Typically, the initiator terminates a transfer upon the successful completion of the transfer. Sometimes, the initiator's bus mastership is relinquished by the bus arbiter (GNT# is removed), often because another device requires bus ownership. This is called initiator preemption and is discussed in later Sections. When the S5933 is an initiator and does not observe a DEVSEL# response to its assertion of FRAME#, it terminates the cycle (master abort).

Normal Cycle Completion

A successful data transfer occurs when both the initiator and target assert their respective ready signals, IRDY# and TRDY#. The last data phase is indicated by the initiator when FRAME# is deasserted during a data transfer. A normal cycle completion occurred if the target does not assert STOP#. Figure 6 shows the signal relationships defining a normal transfer completion.

Figure 5. Single Data Phase PCI Bus Write of S5933 Registers (S5933 as Target)

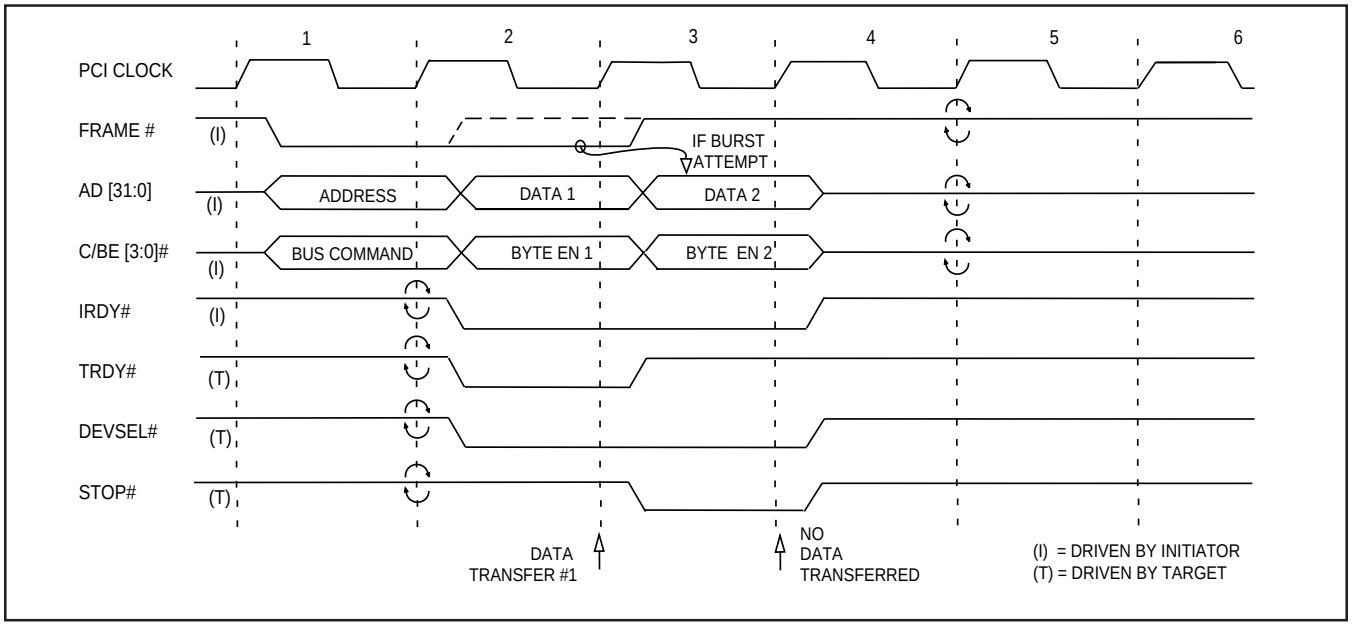
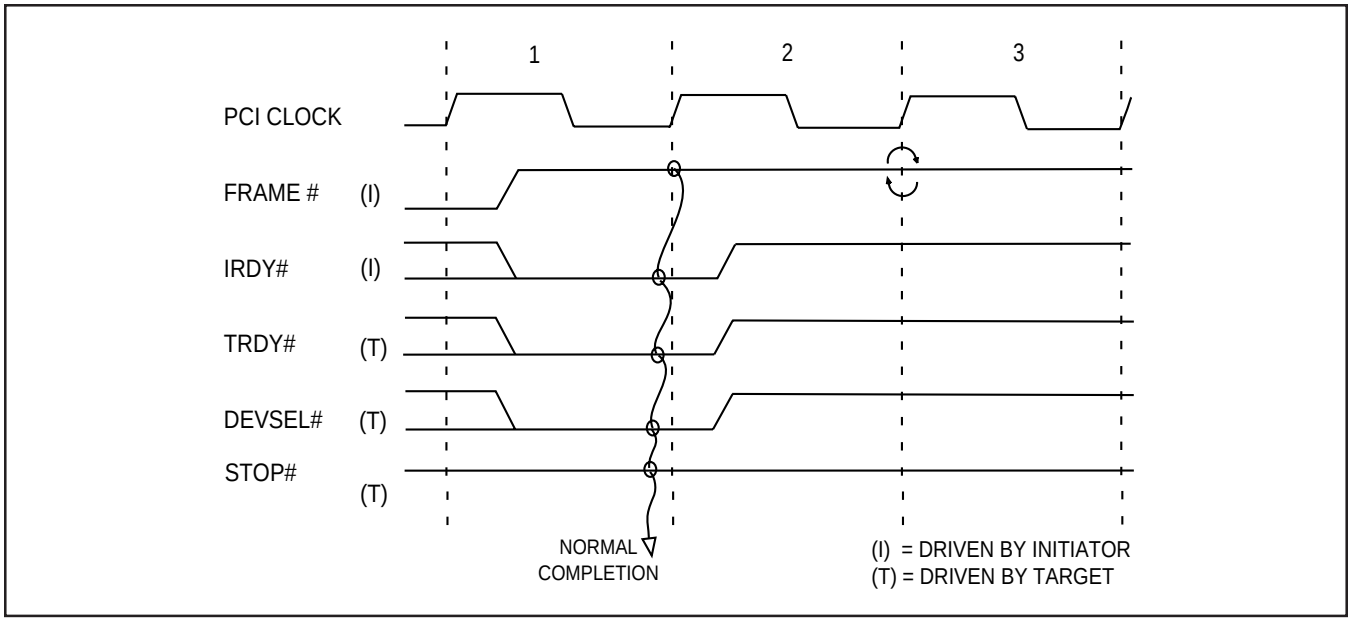


Figure 6. Master-Initiated, Normal Completion (S5933 as either Target or Initiator)



Initiator Preemption

A PCI initiator (bus master) is said to be preempted when the system platform deasserts the initiator's bus grant signal, GNT#, while it still requests the bus (REQ# asserted). This situation occurs if the initiator's latency timer expires and the system platform (bus arbitrator) has a bus master request from another device. The S5933 Master Latency Timer register controls the S5933 responsiveness to the removal of a bus grant (preemption). The presence of a Master Latency Timer register can cause two preemption situations:

1) Removal of GNT# when the latency timer is non-zero (S5933 is guaranteed to still "own the bus").
2) Removal of the GNT# after the latency timer has expired.

The first situation is depicted in Figure 7, when the latency timer has not expired. Preemption with a zero or expired latency timer is shown in Figure 8.

Figure 7. Master Initiated Termination Due to Preemption and Latency Timer Active (S5933 as Master)

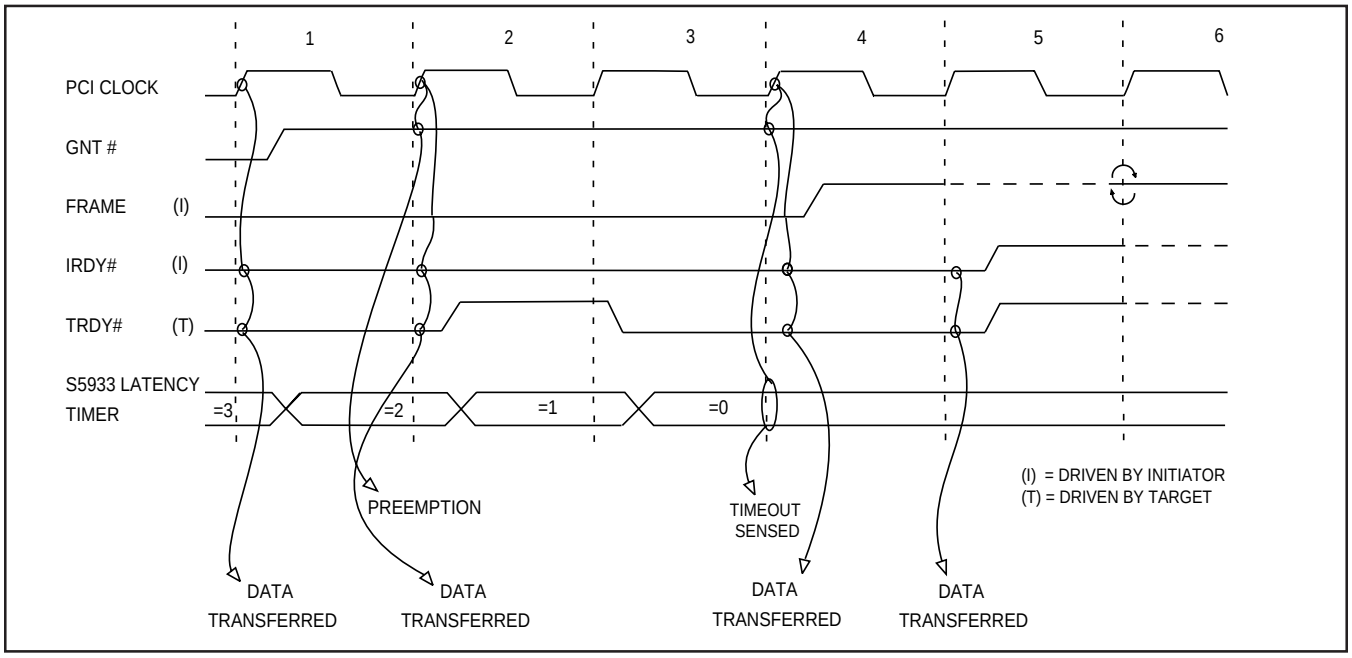
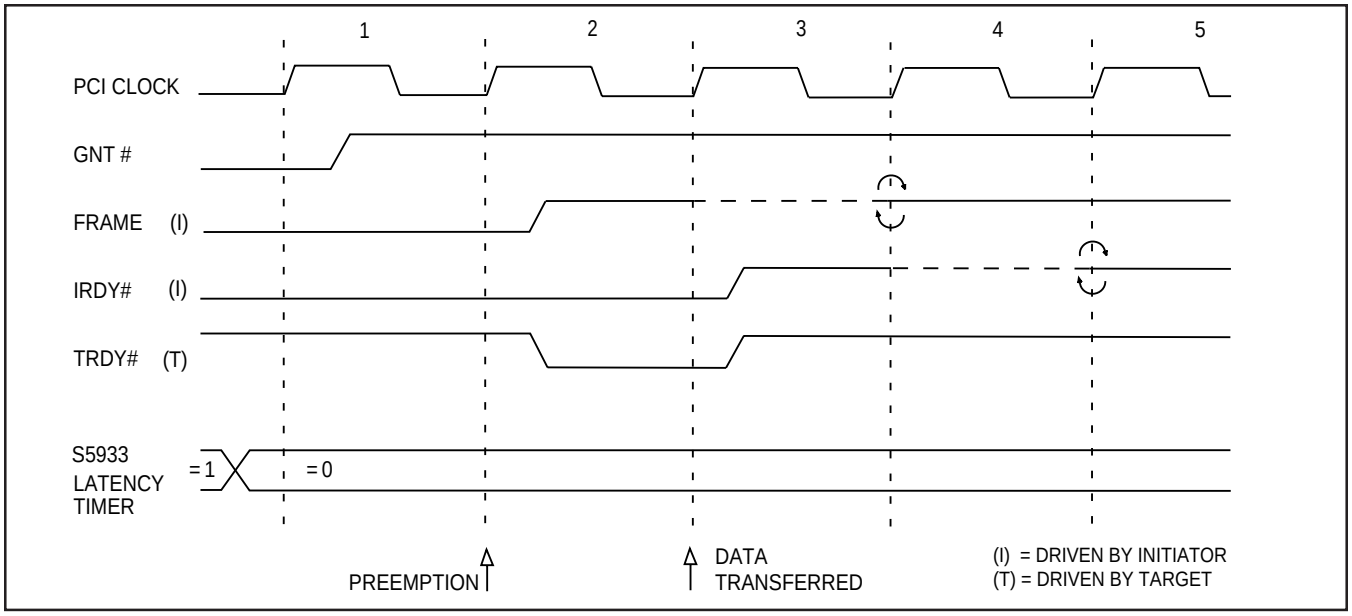


Figure 8. Master Initiated Termination Due to Preemption and Latency Timer Expired (S5933 as Master)



Master Abort

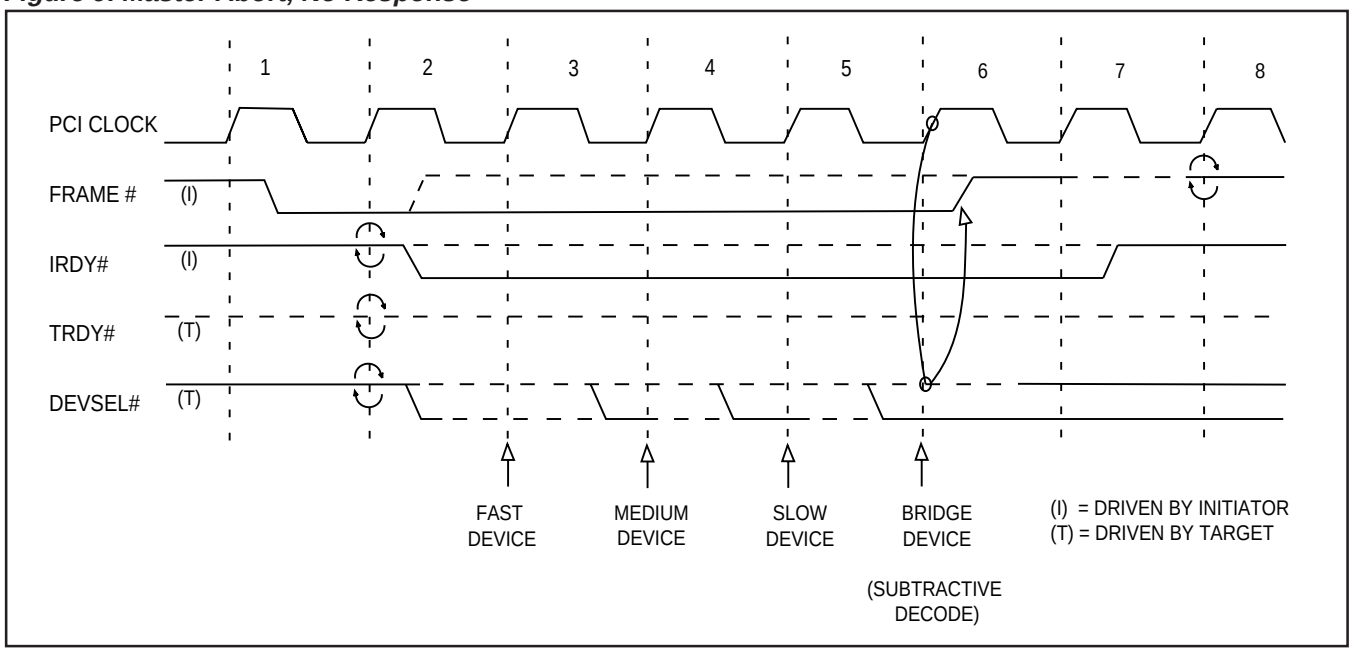
PCI accesses to nonexistent or disabled targets never observe DEVSEL# being asserted. In this situation, it is necessary for the initiator to abort the transaction (master abort). As an initiator, S5933 waits for six clock periods after asserting FRAME# to determine whether a master abort is required. These six clock periods allow slow targets, which may require several bus clocks before being able to assert DEVSEL#, to respond. It is also possible a PCI bridge device is present which employs “subtractive” decoding. A device which does a subtractive decode asserts DEVSEL#, claiming the cycle, when it sees that no other device has asserted it three clocks after the address phase.

If DEVSEL# is not asserted, the S5933 deasserts FRAME# (if asserted) upon the sixth clock period (Figure 9). IRDY# is deasserted by the S5933 during the next clock. The occurrence of a master abort causes the S5933 to set bit 13 (Master Abort) of the PCI Status Register, indicating an error condition.

Target-Initiated Termination

There are situations where the target may end a transfer prematurely. This is called “target-initiated termination.” Target terminations fall into three categories: disconnect, retry, and target abort. Only the disconnect termination completes a data transfer.

Figure 9. Master Abort, No Response



Target Disconnects

There are many situations where a target may disconnect. Slow responding targets may disconnect to permit more efficient (faster) devices to be accessed while they prepare for the next data phase, or a target may disconnect if it recognizes that the next data phase in a burst transfer is out of its address range. A target disconnects by asserting STOP#, TRDY#, and DEVSEL# as shown in Figures 10a and 10b. The initiator in Figure 10a responds to the disconnect condition by deasserting FRAME# on the following clock but does not complete the data transfer until IRDY# is asserted. This situation can only occur when the S5933 is a target. When the S5933 is an initiator, IRDY# is always asserted during the data phase (no initiator wait states). The timing diagram in Figure 10b applies to the S5933 as either a target disconnecting or an initiator with its target performing a disconnect. The S5933 performs a target disconnect if a burst access is attempted to the PCI Operation Registers.

Figure 10a. Target Disconnect Example 1 (IRDY# deasserted)

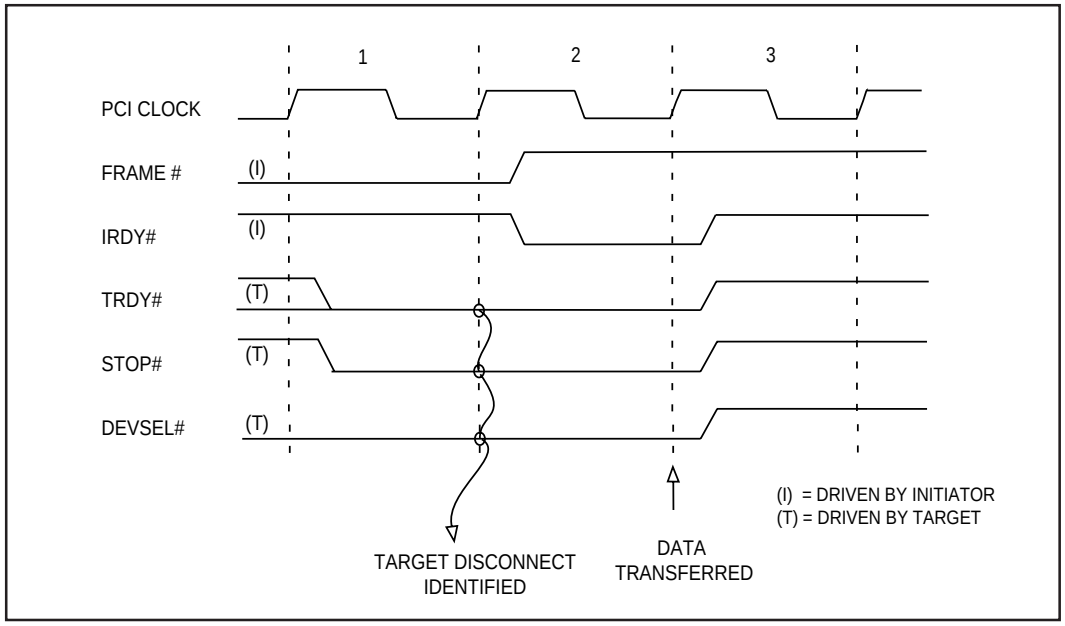
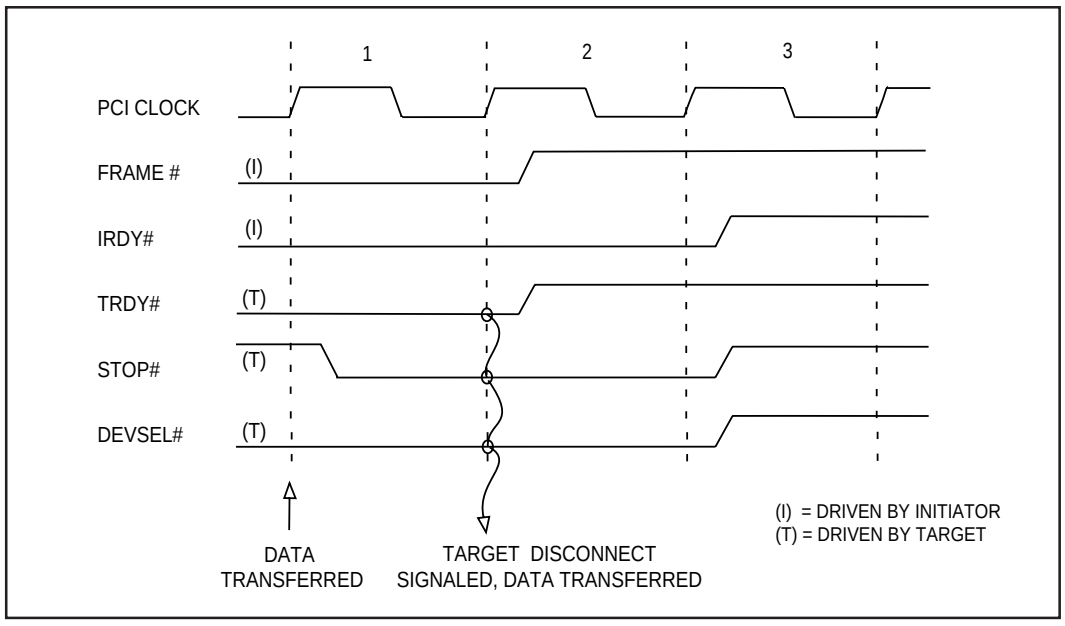


Figure 10b. Target Disconnect Example 2 (IRDY# asserted)



Target Requested Retries

When the S5933 FIFO registers are accessed (S5933 as a target) and data is unavailable (empty FIFO) for read transfers or cannot be accepted for write transfers (full FIFO), the S5933 immediately terminates the cycle by requesting a retry. The S5933 also initiates a retry for Pass-Thru writes where the Add-On has not completed the preceding Pass-Thru write by asserting PTRDY#, and for Pass-Thru reads where the Add-On cannot supply data within 8 PCI clocks (16 clocks for the first data phase of a burst). A retry is requested by a target asserting both STOP# and DEVSEL# while TRDY# is deasserted. Figure 11 shows the behavior of the S5933 when performing a target-initiated retry.

Target Aborts

A target abort termination represents an error condition where no number of retries will produce a successful target access. A target abort is uniquely identified by the target deasserting DEVSEL# and TRDY# while STOP# is asserted. When a target performs an abort, it must also set bit 11 of its PCI Status register. The S5933 configuration and operation registers never respond with a target abort when accessed. If the S5933 encounters this condition when operating as a PCI initiator, the S5933 sets bit 12 (received target abort) in the PCI Status register. Figure 12 depicts a target abort cycle.

Target termination types are summarized in Table 2.

Figure 11. Target-Initiated Retry

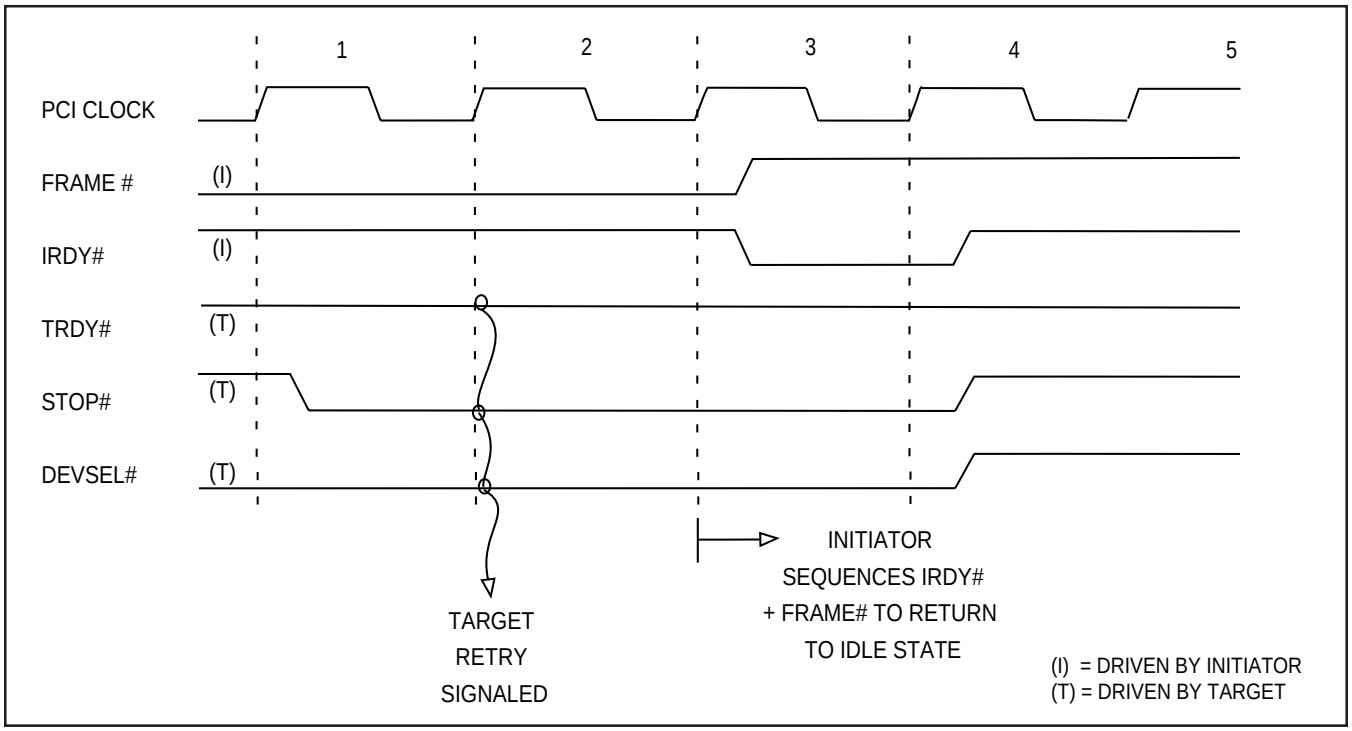


Table 2. Target Termination Types

Termination	DEVSEL#	STOP#	TRDY#	Comment
Disconnect	on	on	on	Data is transferred. Transaction needs to be re-initiated to complete.
Retry	on	on	off	Data was not transferred. Transaction should be tried later.
Abort	off	on	off	Data was not transferred. Fatal error.

Figure 12. Target Abort Example

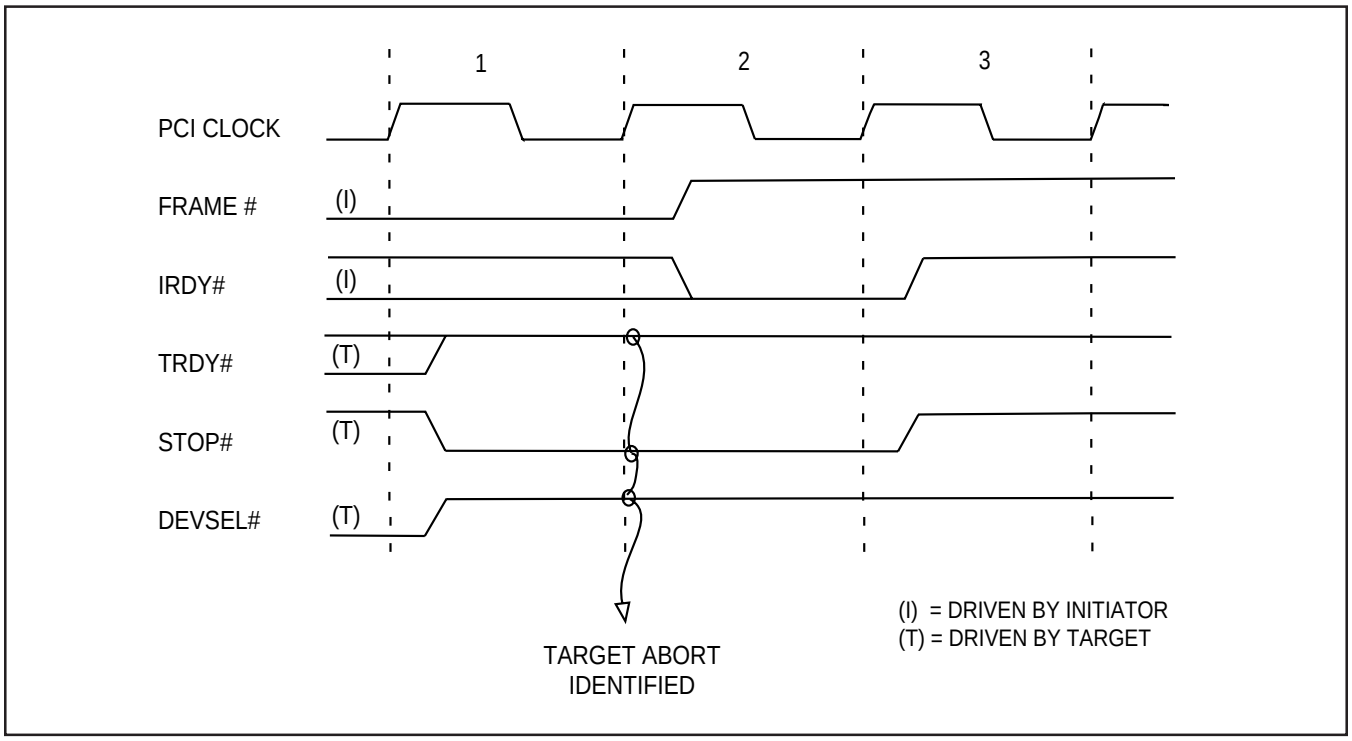
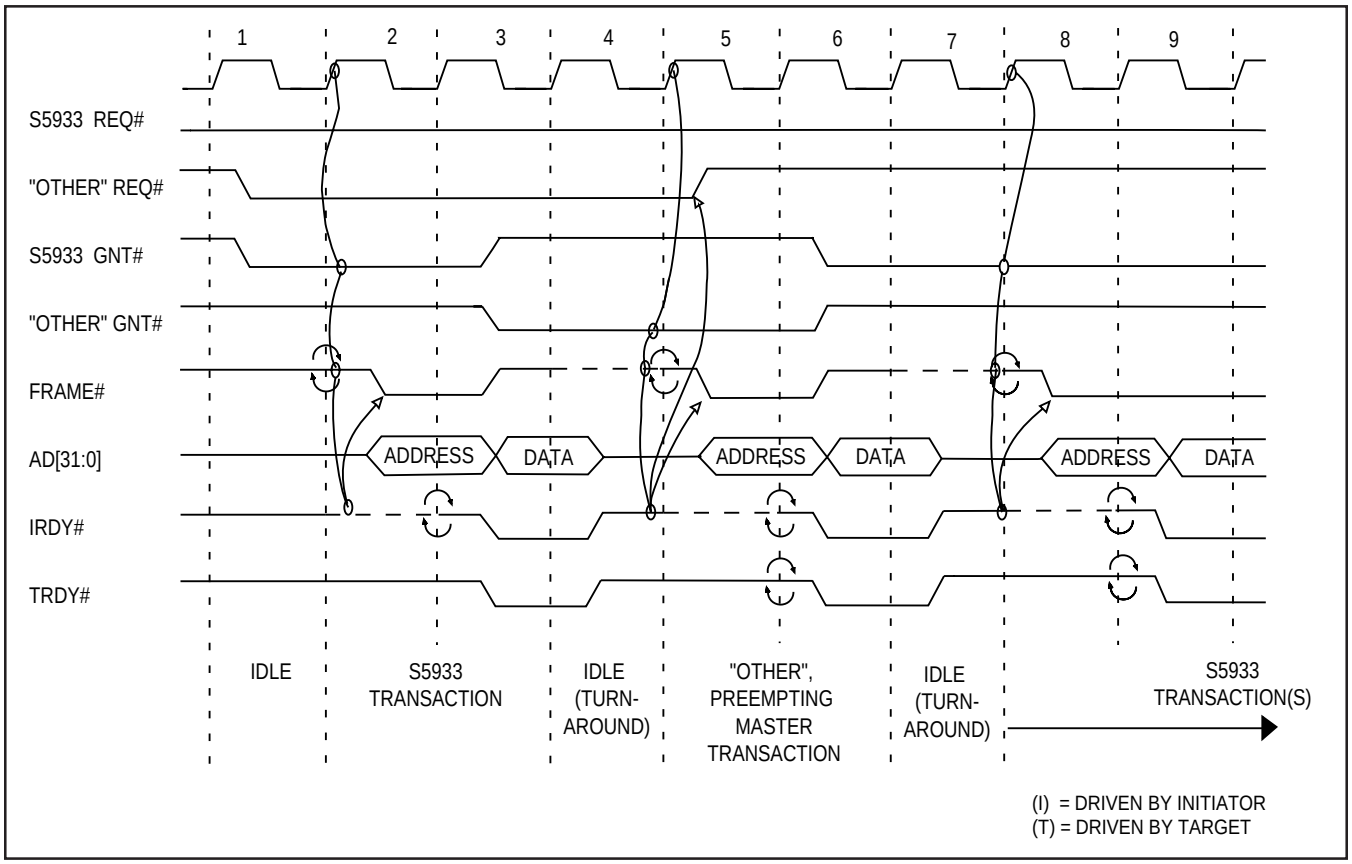


Figure 13. PCI Bus Arbitration and S5933 Bus Ownership Example



PCI BUS MASTERSHIP

When the S5933 requires PCI bus mastership, it presents a request via the REQ# signal. This signal is connected to the system's PCI bus arbiter.

Only one initiator (bus master) may control the PCI bus at a given time. The bus arbiter determines which initiator is given control of the bus. Control is granted to a requesting device by the arbiter asserting that device's grant signal (GNT#). Each REQ#/GNT# signal pair is unique to a given PCI agent.

After asserting REQ#, the S5933 assumes bus ownership on the first PCI clock edge where its GNT# input is asserted along with FRAME# and IRDY# deasserted (indicating no other device is generating PCI bus cycles). Once ownership is established by the S5933, it maintains ownership as long as the arbiter keeps its GNT# asserted. If GNT# is deasserted, the S5933 completes the current transaction. The S5933 does this by deasserting FRAME# and then deasserting IRDY# upon data transfer. Figure 13 shows a sequence where the S5933 is granted ownership of the bus and then is preempted by another master before the S5933 can finish its current transaction.

Bus Mastership Latency Components

It is often necessary for system designers to predict and guarantee that a minimum data transfer rate is sustainable to support a particular application. In the design of a bus mastering application, knowledge of the maximum delay a device might encounter from the time it requests the PCI bus to the time in which it is actually granted the bus is desirable. This allows the design to provide adequate data buffering. The PCI specification refers to this bus request to grant delay as "arbitration latency."

Once a PCI initiator has been granted the bus, the PCI specification defines the delay from the grant to the new initiator's assertion of FRAME# as the "bus acquisition latency." Afterwards, the delay from FRAME# asserted to target ready (TRDY#) asserted is defined as "target latency." Figure 14 shows a time-line depicting the components of PCI bus access latency.

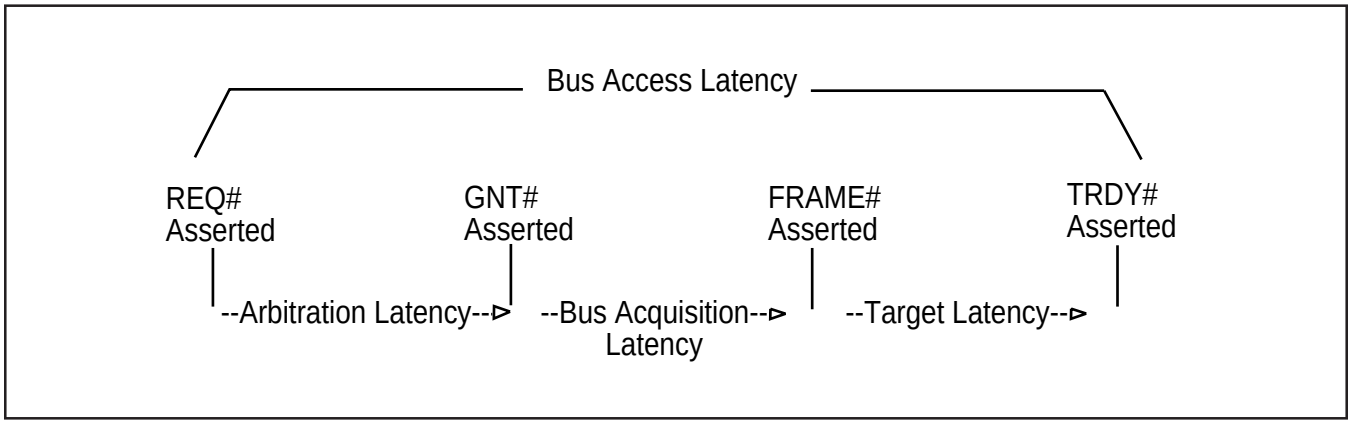
There are numerous configuration variations possible with the PCI specification. A system designer can determine whether a bus master can support a critical, timely transfer by establishing a specific configuration and by defining these latency values. The S5933, as an initiator, produces the fastest response allowable for its bus acquisition latency (GNT# to FRAME# asserted). The S5933 also implements the PCI Master Latency Timer. Once granted the bus, the S5933 is guaranteed ownership for a minimum amount of time defined by the Master Latency Timer. The S5933, as an initiator, cannot control the responsiveness of a particular target nor the bus arbitration delay.

The PCI specification provides two mechanisms to control the amount of time a master may own the bus. One mechanism is through the master (master-initiated termination). The other is by the target and is achieved through a target-initiated disconnect.

Bus Arbitration

Although the PCI specification defines the condition that constitutes bus ownership, it does not provide rules to be used by the system's PCI bus arbiter in deciding which master is to be granted the PCI bus next. The arbitration priority scheme implemented by a system may be fixed, rotational, or custom. The arbitration latency is a function of the system, not the S5933.

Figure 14. PCI Bus Access Latency Components



Bus Acquisition

Once GNT# is asserted, giving bus ownership to the S5933, the S5933 must wait until the PCI bus becomes idle. This delay is called bus acquisition latency and involves the state of the signals FRAME# and IRDY#. The current bus master must complete its current transaction before the S5933 may drive the bus. Table 3 depicts the four possible combinations of FRAME# and IRDY# with their interpretation.

Target Latency

The PCI specification requires that a selected target relinquish the bus should an access to that target require more than eight PCI clock periods (16 clocks for the first data phase in a burst). Slow targets can exist within the PCI specification by using the target initiated retry. This prevents slow target devices from potentially monopolizing the PCI bus and also allows more accurate estimations for bus access latency.

Target Locking

It is possible for a PCI bus master to obtain exclusive access to a target ("locking") through use of the PCI bus signal LOCK#. LOCK# is different from the other PCI bus signals because its ownership may belong to any bus master, even if it does not currently have ownership of the PCI bus. The ownership of LOCK#, if not already claimed by another master, may be achieved by the current PCI bus master on the clock period following the initial assertion of FRAME#. Figure 15 describes the signal relationship for establishing a lock. The ownership of LOCK#, once established, persists even while other bus masters control the bus. Ownership can only be relinquished by the master which originally established the lock.

Figure 15. Engaging the LOCK# Signal

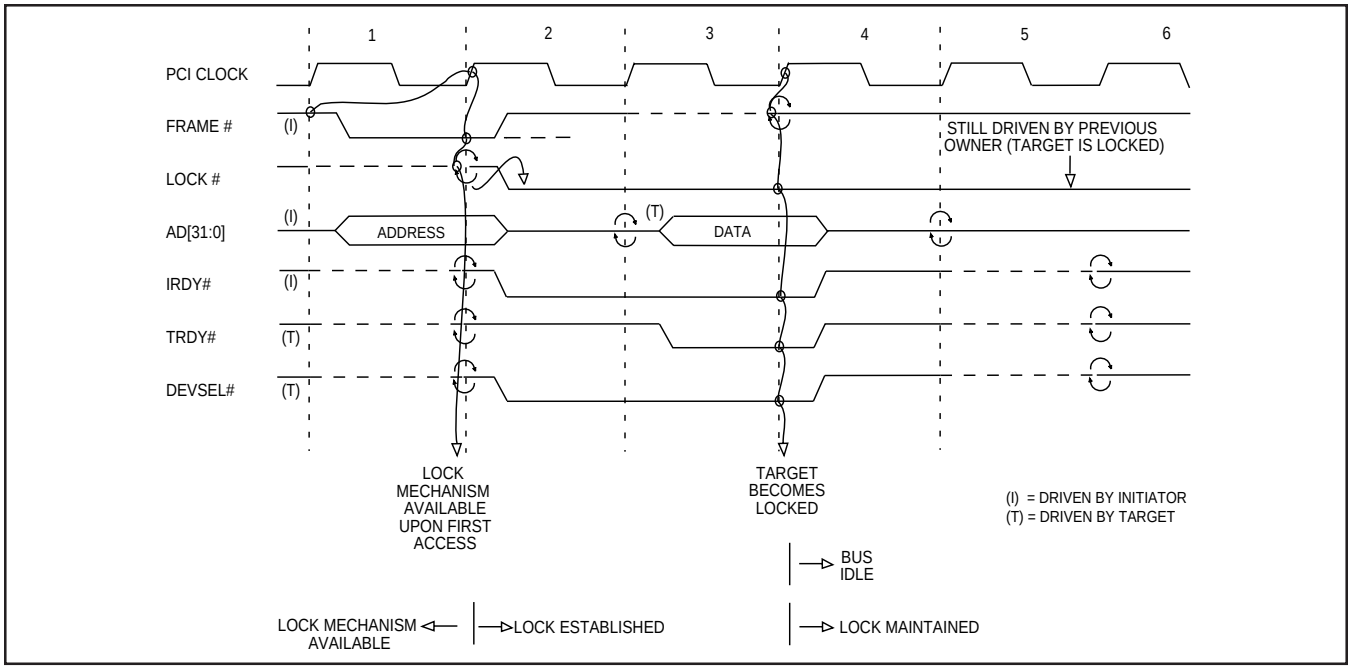


Table 3. Possible Combinations of FRAME# and IRDY#

FRAME#	IRDY#	Description
deasserted	deasserted	Bus Idle
deasserted	asserted	The initiator is ready to complete the last data transfer of a transaction.
asserted	deasserted	An Initiator has a transaction in progress but is not able to complete the data transfer on this clock.
asserted	asserted	An initiator has a transaction in progress and is able to complete a data transfer.

Targets selected with LOCK# deasserted during the assertion of FRAME# (clock period 1 of Figure 15), which encounter the assertion of LOCK# during the following clock (clock period 2 of Figure 15) are thereafter considered “locked.” A target, once locked, requires that subsequent accesses to it deassert LOCK# while FRAME# is asserted. Figure 16 show a subsequent access to a locked target by the master which locked it. Because LOCK# is owned by a single master, only that master is able to deassert it at the beginning of a transaction (allowing successful access to the locked target). A locked target can only be unlocked during the clock period following the last data transfer of a transaction when the LOCK# signal is deasserted.

An unlocked target ignores LOCK# when it observes that LOCK# is already asserted during the first clock period of a transaction. This allows other masters to access other (unlocked) targets. If an access to a locked target is attempted by a master other than the one that locked it, the target responds with a retry request, as shown in Figure 17.

The S5933 responds to and supports bus masters which lock it as a target. When the S5933 is a bus master, it never attempts to lock a target, but it honors a target’s request for retry if that target is locked by another master.

Figure 16. Access to a Locked Target by its Owner

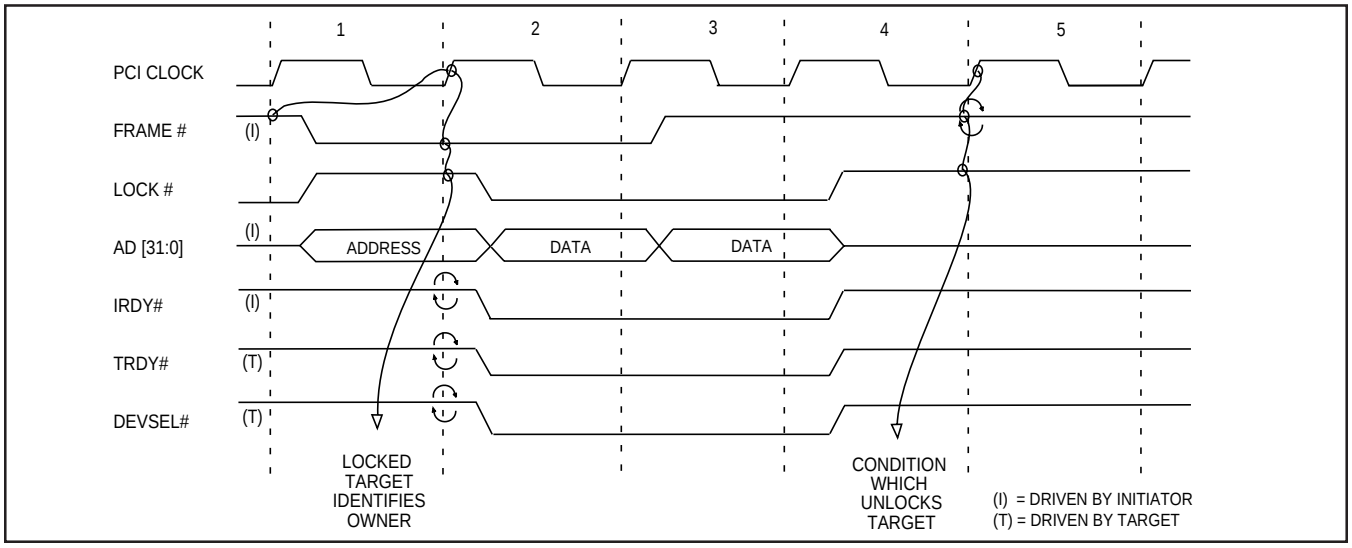
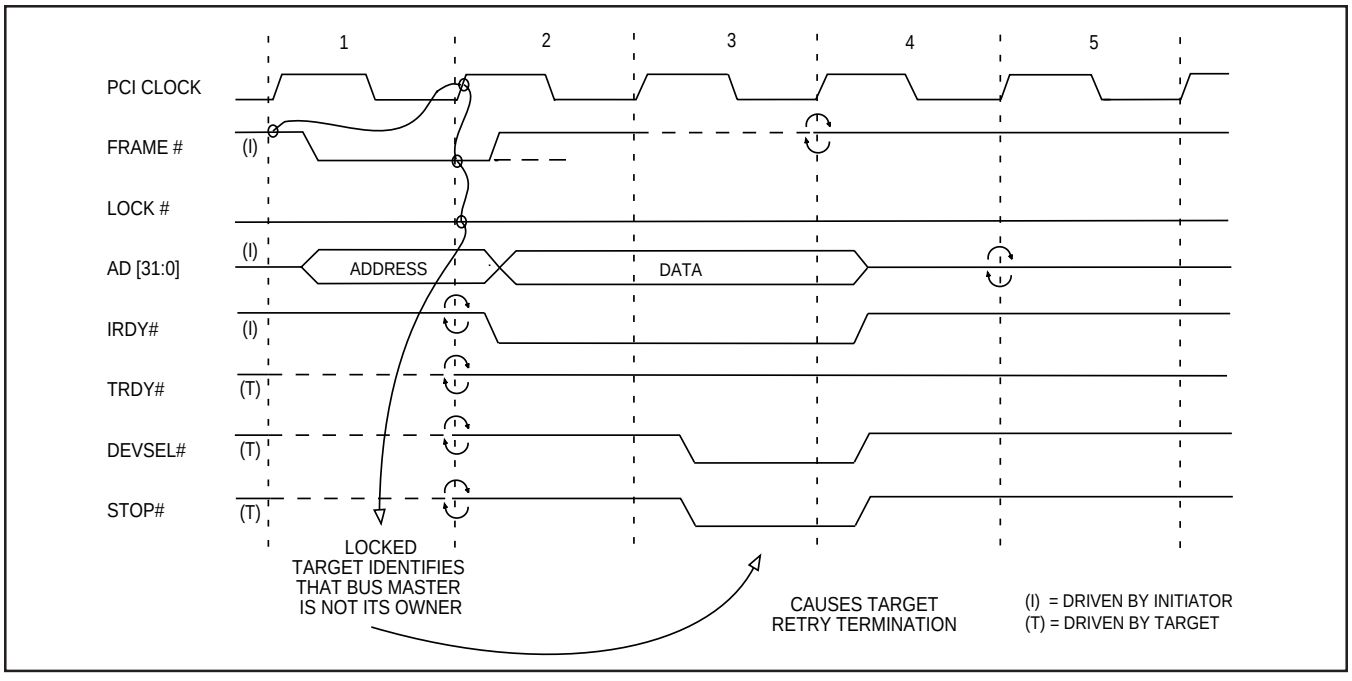


Figure 17. Access Attempt to a Locked Target



PCI BUS INTERRUPTS

The S5933 controller is able to generate PCI bus interrupts by asserting the PCI bus interrupt signal (INTA#). INTA# is a multisourced, wire-ORed signal on the PCI bus and is driven by an open drain output on the S5933. The assertion and deassertion of INTA# have no fixed timing relationship with respect to the PCI bus clock. Once the S5933 asserts INTA#, it remains asserted until the interrupt source is cleared by a write to the Interrupt Control/Status Register (INTCSR).

PCI BUS PARITY ERRORS

The PCI specification defines two error-reporting signals, PERR# and SERR#. These signals indicate a parity error condition on the signals AD[31:0], C/BE[3:0]#, and PAR. The validity of the PAR signal is delayed one clock period from its corresponding AD[31:0] and C/BE[3:0]# signals. Even parity exists when the total number of ones in the group of signals is equal to an even number. PERR# is the error-reporting mechanism for parity errors that occur during the data phase for all but PCI Special Cycle commands. SERR# is the error-reporting mechanism for parity errors that occur during the address phase.

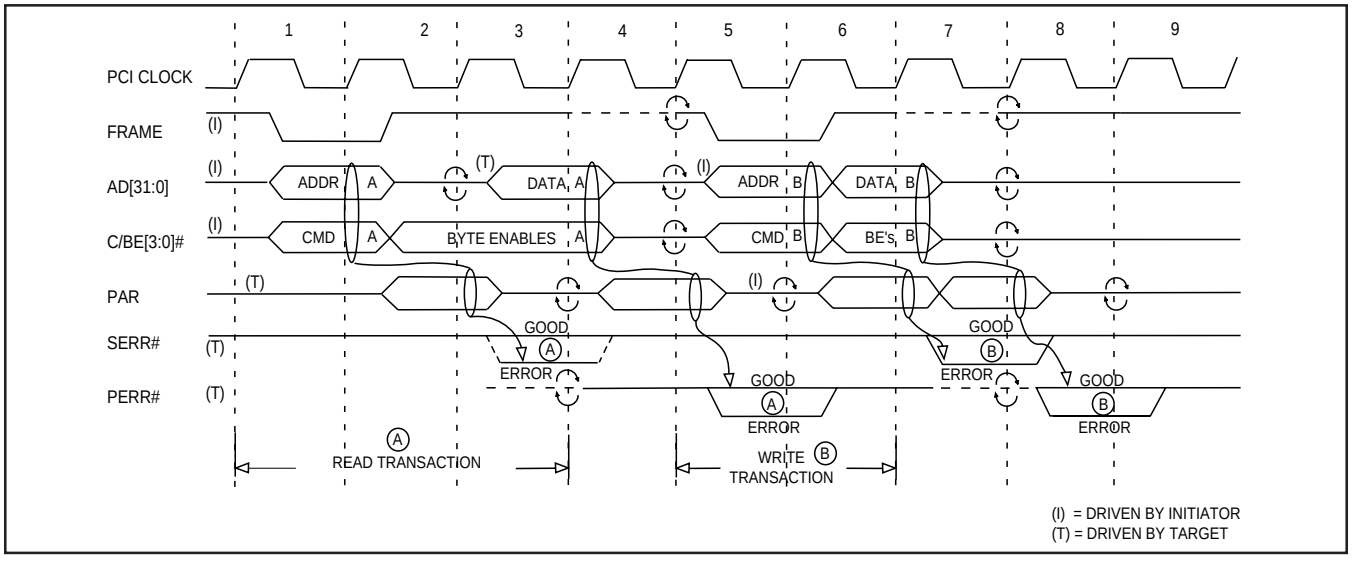
The timing diagram in Figure 18 shows the timing relationships between the signals AD[31:0], C/BE[3:0]#, PAR, PERR# and SERR#.

The S5933 asserts SERR# if it detects odd parity during an address phase, if enabled. The SERR# enable bit is bit 8 in the S5933 PCI Command Register. The odd parity error condition involves the state of signals AD[31:0] and C/BE[3:0]# when FRAME# is first asserted and the PAR signal during the following clock. If an error is detected, the S5933 asserts SERR# on the following (after PAR valid) clock. Since many targets may observe an error on an address phase, the SERR# signal is an open drain multisourced, wire-ORed signal on the PCI bus. The S5933 drives SERR# low for one clock period when an address phase error is detected. Once an SERR error is detected by the S5933, the PCI Status register bit 14, System Error, is set and remains until cleared through software or a hardware reset.

The PERR# signal is similar to the SERR# with two differences: it reports errors for the data phase and is only asserted by the device receiving the data. The S5933 drives this signal (removed from tri-state) when it is the selected target for write transactions or when it is the current master for bus read transactions. The parity error conditions are only reflected by the PERR# pin if the Parity Error Enable bit (bit 6) of the PCI Command register is set. Upon the detection of a data parity error, the Detected Parity Error bit (bit 15) of the PCI Status Register is set. Unlike the PERR# signal pin, this Status bit sets regardless of the state of the PCI Command register Parity Error Enable bit. An additional status bit (bit 8) called “Data Parity Reported” of the PCI Status register is employed to report parity errors that occur when the S5933 is the bus master. The “Data Parity Error Reported” status requires that the Parity Error Enable bit be set in the PCI Command register.

The assertion of PERR# occurs two clock periods following the data transfer. This two-clock delay occurs because the PAR signal does not become valid until the clock following the transfer, and an additional clock is provided to generate and assert PERR# once an error is detected. PERR# is only asserted for one clock cycle for each error sensed. The S5933 only qualifies the parity error detection during the actual data transfer portion of a data phase (when both IRDY# and TRDY# are asserted).

Figure 18. Error Reporting Signals



ADD-ON BUS INTERFACE

This chapter describes the Add-On bus interface for the S5933. The S5933 is designed to support connection to a variety of microprocessor buses and/or peripheral devices. The Add-On interface controls S5933 operation through the Add-On Operation Registers. These registers act as the Pass-Thru, FIFO, non-volatile memory and mailbox interfaces as well as offering control and status information.

Depending on the register being accessed, the interface may be synchronous, asynchronous, or configurable. To enhance performance and simplify Add-On logic design, some registers allow direct access with a single device input pin. The following sections describe the various interfaces to the PCI bus and how they are accessed from the Add-On interface.

ADD-ON OPERATION REGISTER ACCESSES

The S5933 Add-On bus interface is very similar to that of a memory or peripheral device found in a microprocessor-based system. A 32-bit data bus with individual read and write strobes, a chip enable and byte enables are provided. Other Add-On interface signals are provided to simplify Add-On logic design.

Accesses to the S5933 registers are done synchronous or asynchronous to BPCLK. For S5933 functions that are compatible with an Add-On microprocessor interface, it is helpful to allow an asynchronous interface, as the processor may not operate at the PCI bus clock frequency.

Add-On Interface Signals

The Add-On interface provides a small number of system signals to allow the Add-On to monitor PCI bus activity, indicate status conditions (interrupts), and allow Add-On bus configuration. A standard bus interface is provided for Add-On Operation Register accesses.

System Signals

BPCLK and SYSRST# allow the Add-On interface to monitor the PCI bus status. BPCLK is a buffered version of the PCI clock. The PCI clock can operate from 0 MHz to 33 MHz. SYSRST# is a buffered version of the PCI reset signal, and may also be toggled by host application software through bit 24 of the Bus Master Control/Status Register (MCSR).

IRQ# is the Add-On interrupt output. This signal is active low and can indicate a number of conditions. Add-On interrupts may be generated from the mailbox or FIFO interfaces. The exact conditions which generate an interrupt are discussed in the mailbox and FIFO chapters. The interrupt output is deasserted when acknowledged by an access to the Add-On Interrupt Control/Status Register (AINT). All interrupt sources are cleared by writing a one to the corresponding interrupt bit.

The MODE input on the Add-On interface configures the datapath width for the Add-On interface. MODE low indicates a 32-bit data bus. MODE high indicates a 16-bit data bus. For 16-bit operation, BE3# is redefined as ADR1, providing an extra address input. ADR1 selects the low or high words of the 32-bit S5933 Add-On Operation Registers.

Register Access Signals

Simple register accesses to the S5933 Add-On Operation Registers take two forms: synchronous to BPCLK and asynchronous. The following signals are required to complete a register access to the S5933.

BE[3:0]# Byte Enable Inputs. These S5933 inputs identify valid byte lanes during Add-On transactions. When MODE is set for 16-bit operation, BE2# is not defined and BE3# becomes ADR1.

ADR[6:2] Address Inputs. These address pins identify the specific Add-On Operation Register being accessed. When configured for 16-bit operation (MODE=1), an additional input, ADR1 is available to allow the 32-bit operation registers to be accessed with two 16-bit cycles.

RD# Read Strobe Input.

WR# Write Strobe Input.

SELECT# Chip Select Input. This input identifies a valid S5933 access.

DQ[31:0] Bidirectional Data Bus. These I/O pins are the S5933 data bus. When configured for 16-bit operation, only DQ[15:0] are valid.

In addition, there are dedicated signals for FIFO accesses (RDFIFO# and WRFIFO#) and Pass-Thru address accesses (PTADR#). These are discussed separately in the FIFO and Pass-Thru sections of this chapter.

The internal interfaces of the S5933 allow Add-On Operation Registers to be accessed asynchronous to BPCLK (synchronous to the rising edge of the read or write strobe). The exception to this is the Add-On General Control/Status Register. This is due to the async nature of FIFO status bits changing as the PCI bus reads data. For Pass-Thru operations, the Pass-Thru Data Register accesses are synchronous to BPCLK to support burst transfers. The FIFO port may also be accessed synchronous to BPCLK, if configured to do so.

Asynchronous Register Accesses

For many Add-On applications, Add-On logic does not operate at the PCI bus frequency. This is especially true for Add-Ons implementing a microprocessor, which may be operating at a lower (or higher) frequency. Figures 1 and 2 show asynchronous Add-On Operation Register accesses. Exact AC timings are detailed in the Electrical and AC Characteristics chapter (Chapter 13).

For asynchronous reads (Figure 1), data is driven on the data bus when RD# is asserted. When RD# is not asserted, the DQ[31:0] outputs float. A valid address and valid byte enables must be presented before correct data is driven. RD# has both a minimum inactive time and a minimum active time for asynchronous accesses.

For asynchronous writes (Figure 2), data is clocked into the S5933 on the rising edge of the WR# input. Address, byte enables, and data must all meet setup and hold times relative to the rising edge or WR#. WR# has both a minimum inactive time and a minimum active time for asynchronous accesses

Synchronous FIFO and Pass-Thru Data Register Accesses

To obtain the highest data transfer rates possible, Add-On logic should operate synchronously with the PCI clock. The buffered PCI clock (BPCLK) is provided for this purpose. A synchronous interface with Pass-Thru mode or the FIFO allows data to be transferred at the maximum PCI bus bandwidth (132 MBytes/sec) by allowing burst accesses with the Add-On interface. The RD# and WR# inputs become enables, using BPCLK to clock data into and out of registers. This section applies only to synchronous accesses to the FIFO (AFIFO) and Pass-Thru Data (APTD) registers.

Figures 3 and 4 show single-cycle, synchronous FIFO and Pass-Thru Operation Register accesses. Exact AC timings are detailed in the Electrical and AC Characteristics chapter.

For synchronous reads (Figure 3), data is driven onto the data bus when RD# (or RDFIFO#) is asserted. When RD# is not asserted, the DQ[31:0] outputs float. The address, byte enable, and RD# inputs must meet setup and hold times relative to the rising edge of BPCLK. Burst reads may be performed by holding RD# low.

For synchronous writes (Figure 4), data is clocked into the register on the rising edge of BPCLK. Address, byte enables, and data must all meet setup and hold times relative to the rising edge or BPCLK. Burst writes may be performed by holding WR# (or WRFIFO#) low. When holding WR# low, data is clocked in on each BPCLK rising edge.

nv Memory Accesses Through the Add-On General Control/Status Register

To access nv memory contents through the Add-On General Control/Status Register (AGCSTS), special considerations must be made. Internally, all nv memory accesses by the S5933 are synchronized to a divided-down version of the PCI bus clock. Because of this, if nv memory accesses are performed through the AGCSTS register, the register access must be synchronized to BPCLK. The rising edge RD# or WR# is still used to clock data, but these inputs along with the address and byte enables are synchronized to BPCLK. Accesses to AGCSTS for monitoring FIFO or mailbox status, etc., may be done asynchronous to BPCLK.

MAILBOX BUS INTERFACE

The mailbox register names may need some clarification. For the Add-On interface, an outgoing mailbox refers to a mailbox sending information to the PCI bus. An incoming mailbox refers to a mailbox receiving information from the PCI bus. An outgoing mailbox on the Add-On interface is, internally, the same as the corresponding incoming mailbox on the PCI interface and vice-versa.

Figure 1. Asynchronous Add-On Operation Register Read

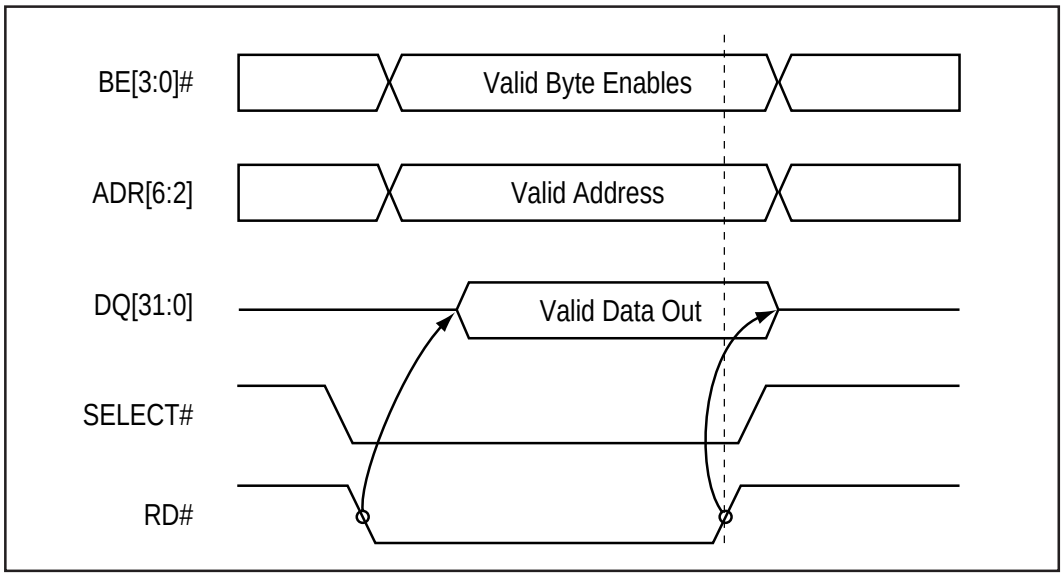


Figure 2. Asynchronous Add-On Operation Register Write

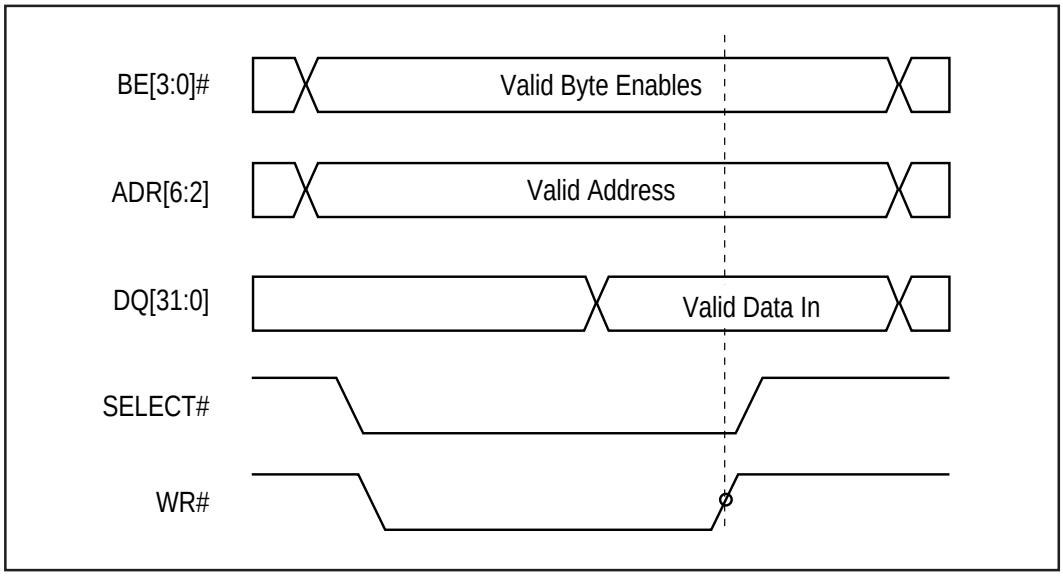


Figure 3. Synchronous FIFO or Pass-Thru Data Register Read

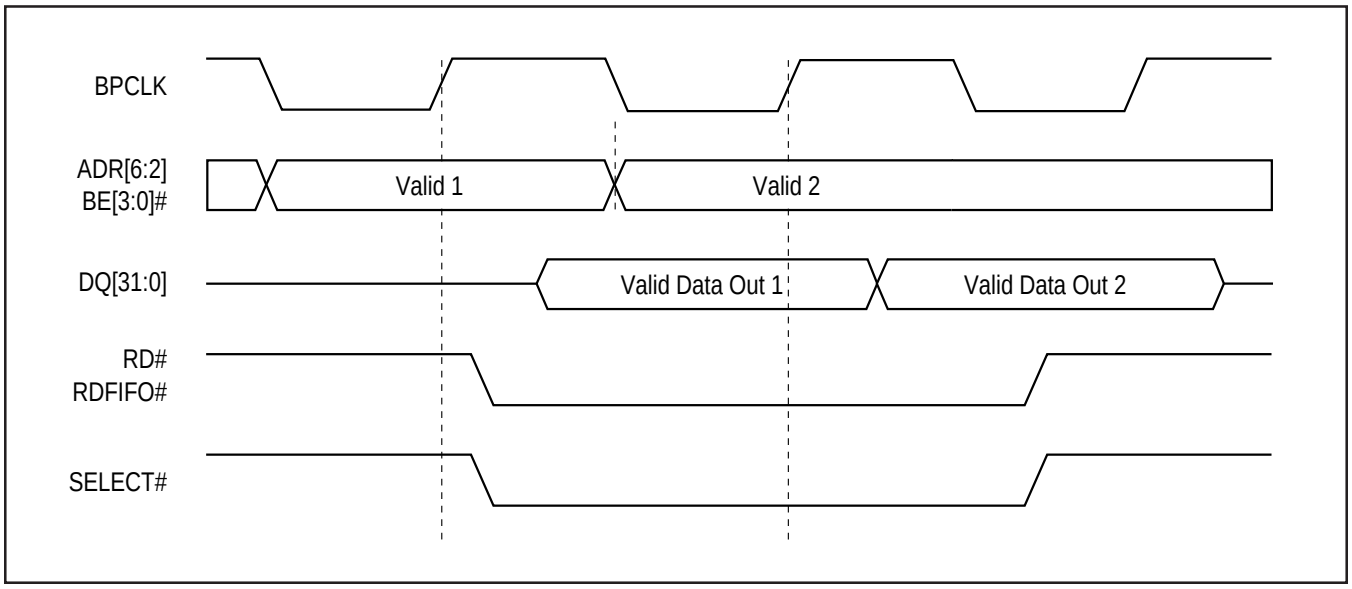
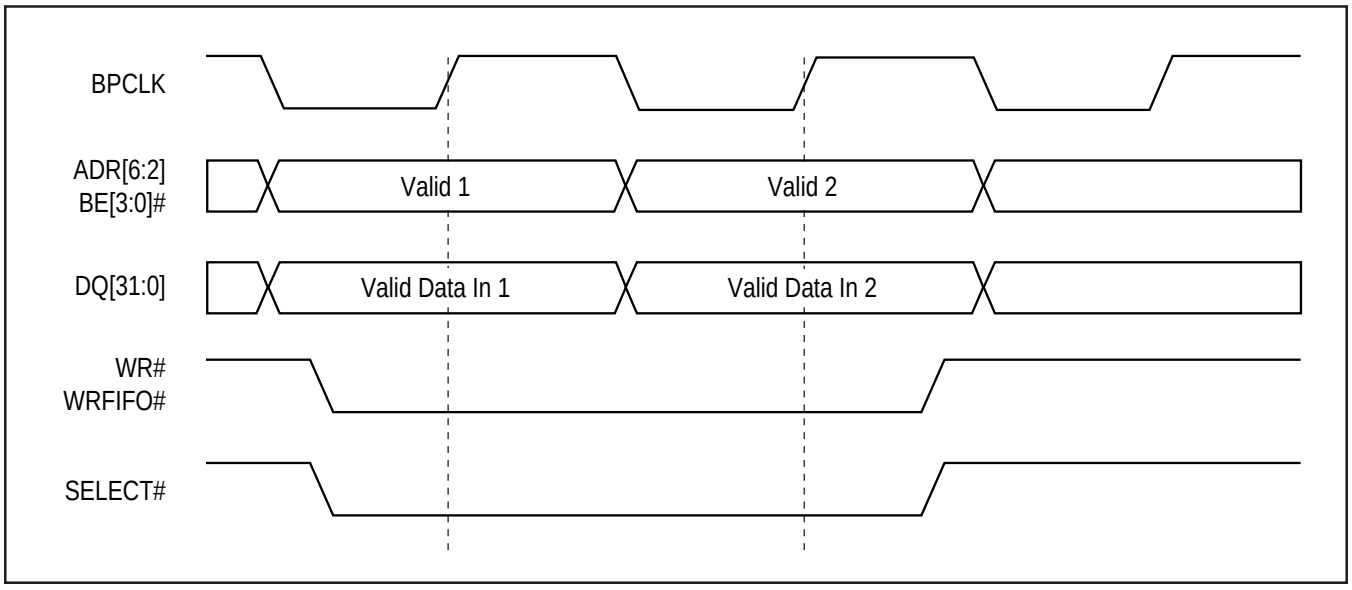


Figure 4. Synchronous FIFO or Pass-Thru Data Register Write



Mailbox Interrupts

Mailboxes can be configured to generate Add-On interrupts (IRQ#) and/or allow the Add-On to generate PCI interrupts (INTA#). Mailbox empty/full status conditions be can used to interrupt the Add-On or PCI host to indicate some action is required. An individual mailbox byte is selected to generate an interrupt when accessed. An outgoing mailbox becoming empty or an incoming mailbox becoming full asserts the interrupt output (if enabled).

When used with a serial nv memory boot device, the mailboxes also provide a way to generate PCI interrupts (INTA#) through hardware. When a serial nv memory boot device is used, the device pin functions EA0 - EA8 are redefined. These pins then provide direct, external access to the Add-On outgoing mailbox 4, byte 3 (which is also PCI incoming mailbox 4, byte 3).

FIFO BUS INTERFACE

The FIFO register on the Add-On interface may be accessed synchronously or asynchronously.Location 45h, bits 6 and 5 in the nv memory boot device determine the interface method. If no boot device is used, the default condition is an asynchronous FIFO interface.

FIFO Direct Access Inputs

RDFIFO# and WRFIFO# are referred to as FIFO 'direct access' inputs. Asserting RDFIFO# is functionally identical to accessing the FIFO with RD#, SELECT#, BE[3:0]#, and ADR[6:2]. Asserting WRFIFO# is functionally identical to accessing the FIFO with WR#, SELECT#, BE[3:0]#, and ADR[6:2]. RD# and WR# must be deasserted when RDFIFO# or WRFIFO# is asserted, but SELECT# may be asserted. These inputs automatically drive the address (internally) to 20h and assert all byte enables. The ADR[6:2] and BE[3:0]# inputs are ignored when using the FIFO direct access inputs. RDFIFO# and WRFIFO# are useful for Add-On designs which cascade an external FIFO into the S5933 FIFO or use dedicated external logic to access the FIFO.

Direct access signals always access the FIFO as 16-bits or 32-bits, whatever the MODE pin is configured for. For 16-bit mode, two consecutive accesses fill or empty the 32-bit FIFO register.

FIFO Status Signals

The FIFO Status signals indicate to the Add-On logic the current state of the S5933 FIFO. A FIFO status change caused by a PCI FIFO access is reflected one PCI clock period after the PCI access is completed (TRDY# asserted). A FIFO status change caused by an Add-On FIFO access is reflected immediately (after a short propagation delay) after the access occurs. For Add-On accesses, FIFO status is updated after the rising edge of BPCLK for synchronous interfaces or after the rising edge of the read or write strobe for asynchronous interfaces.

FIFO Control Signals

For Add-On initiated PCI bus mastering, the FIFO status reset controls FWC# (Add-On to PCI FIFO clear) and FRC# (PCI to Add-On FIFO clear) are available. FWC# and FRC# must be asserted for a minimum of one BPCLK period to be recognized. These inputs are sampled at the rising edge of BPCLK. These inputs should not be asserted unless the FIFO is idle. Asserting a FIFO status reset input during a PCI or Add-On FIFO access results in indeterminate operation.

For Add-On initiated bus master transfers, AMREN (Add-On bus master read enable) and AMWEN (Add-On bus master write enable) are used, in conjunction with the appropriate FIFO status signals, to enable the S5933 to assert its PCI bus request (REQ#).

PASS-THRU BUS INTERFACE

The S5933 Pass-Thru interface is synchronous. The Add-On Pass-Thru Address (APTA) and Add-On Pass-Thru Data (APTD) registers may be accessed pseudo-synchronously.

Although BPCLK is used to clock data into and out of the Pass-Thru registers, accesses may be performed asynchronously. For reads, APTA or APTD data remains valid as long as RD# (or PTADR#) is asserted. A new value is not driven until PTRDY# is asserted by Add-On logic. For writes to APTD, data is clocked into the S5933 on every BPCLK rising edge, but is not passed to the PCI bus until PTRDY# is asserted. PTRDY# must by synchronized to BPCLK.

Pass-Thru Status Indicators

The Pass-Thru status indicators indicate that a Pass-Thru access is in process and what action is required by the Add-On logic to complete the access. All Pass-Thru status indicators are synchronous with the PCI clock.

Pass-Thru Control Inputs

Some Pass-Thru implementations may require an address corresponding to the Pass-Thru data. The Add-On Pass-Thru Address Register (APTA) contains the PCI address for the Pass-Thru cycle. To allow access to the Pass-Thru address without generating an Add-On read cycle, PTADR# is provided. PTADR# is a direct access input for the Pass-Thru address. Asserting PTADR# is functionally identical to accessing the Pass-Thru address register with RD#, SELECT#, BE[3:0]#, and ADR[6:2]. RD# and WR# must be deasserted when PTADR# is asserted, but SELECT# may be asserted. These inputs automatically drive the address (internally) to 28h and assert all byte enables. The ADR[6:2] and BE[3:0]# are ignored when using the PTADR# direct access input. When PTADR# is asserted, the contents of the APTA register are immediately driven onto the Add-On data bus.

The PTADR# direct access signal accesses the Pass-Thru address register as 16-bits or 32-bits, whatever the MODE pin is configured for. For 16-bit mode, PTADR# only presents the lower 16-bits of the APTA register.

PTRDY# indicates that the Add-On has completed the current Pass-Thru access. Multiple Add-On reads or writes may occur to the Pass-Thru data (APTD) register before asserting PTRDY#. This may be required for 8-bit or 16-bit Add-On interfaces using multiple accesses to the 32-bit Pass-Thru data register. In some cases, the Add-On bus may be 32-bits, but logic may require multiple BPCLK periods to read or write data. In this situation, accesses may be extended by holding off PTRDY#. PTRDY# must be synchronized to BPCLK.

NON-VOLATILE MEMORY INTERFACE

The S5933 allows read and write access to the nv memory device used for configuration. Reads are necessary during device initialization as configuration information is downloaded into the S5933. If an expansion BIOS is implemented in the nv memory, the host transfers (shadows) the code into system DRAM. Writes are useful for in-field updates to expansion BIOS code. This allows software to update the nv memory contents without altering hardware.

Non-Volatile Memory Interface Signals

For serial nv memory devices, there are only two signals used to interface with nv memory. SCL is the serial clock, and SDA is the serial data line. The functionality of these signals is described in-detail in the PIN description Section of this book. The designer does not need to generate the timings for SCL and SDA. The S5933 automatically performs the correct serial access when programmed for for serial devices.

For byte-wide nv memory devices, there is an 8-bit data bus (EQ7:0), and a 16-bit address bus (EA15:0) dedicated for the nv memory interface. When a serial nv memory is implemented, many of these pins have alternate functions. The S5933 also has read (ERD#) and write (EWR#) outputs to drive the OE# and WR# inputs on a byte-wide nv memory. The designer does not need to generate the timings for these outputs. The S5933 automatically performs the read and write accesses when programmed for byte wide devices.

Accessing Non-Volatile Memory

The nv memory, if implemented, can be accessed through the PCI interface or the Add-On interface. Accesses from both the PCI side and the Add-On side must be synchronous with the PCI clock (BPCLK for the Add-On). Accesses to the nv memory from the PCI interface are through the Bus Master Control/Status Register (MCSR) PCI Operation Register. Accesses to the nv memory from the Add-On interface are through the Add-On General Control/Status Register (AGCSTS) Add-On Operation Register.

Accesses to the MCSR register are from the PCI bus and are, therefore, automatically synchronous to the PCI clock. Accesses to the AGCSTS register from the Add-On side must be synchronous with respect to BPCLK.

Some nv memories may contain Expansion ROM BIOS code for use by the host software. During initialization, the Expansion BIOS is located within system memory. The starting location of the nv memory is stored in the Expansion ROM Base Address Register in the S5933 PCI Configuration Registers. A PCI read from this region results in the S5933 performing four consecutive byte access to the nv memory device. Writes to the nv memory are not allowed by writing to this region. Writes to the nv memory must be performed as described below.

The S5933 contains two latches within the MCSR register to control and access the NVRAM. One is an 8 bit latch called the NVRAM Address/Data Register which is used to hold NVRAM address and data information. The other is a 3 bit latch called the NVRAM Access Control Register which is used to direct the address and data information and to control the NVRAM itself. Reading or writing to the NVRAM is performed through bits D31:29 of this register. These bits are enable and decode controls rather than a command or instruction to be executed. D31 of this register is the primary enable bit which allows all accesses to occur. When written to a ‘1’, D31 enables the decode bits D30 and D29 to direct the data contained in the address/data latch, D23:16, to the low address, high address or data latches. D31 should be thought of as “opening a door” where as long as D31 = 1, then the door is open for address or data information to be altered. The table on page 5-16 of the S5933 data book shows the D31:29 bit combinations for reading, writing, and loading address/data information. Additionally, D31 doubles as an S5933 status bit. A ‘1’ indicates that the S5933 is currently busy reading or writing to the NVRAM. A ‘0’ indicates a complete or inactive state.

For the examples below, we will assume the S5933 is I/O mapped with a base address of FC00h. These examples will read one byte of the Vendor ID and write one byte to the Vendor ID.

This example will write 1 byte from NVRAM location 0040h and read it back:

In	**FC00h + 3Fh** (offset of NVRAM Access Control Register) until **D31 = 0** (not busy).
Out	**FC00h + 3FH** an **80h** (CMD to load the low address byte). This sets decode bits and opens door for low address latch.
Out	**FC00h + 3Eh** (offset of Address/Data Register) **40h** (the low byte of the address desired) 40h goes into latch but is not latched yet.
Out	**FC00h + 3Fh** an **A0h** (CMD to load the high address byte). This latches the low address through changing the decode bits and opens the door for the high address latch.
Out	**FC00h + 3Eh** a **00h** (the high byte of the address desired). 00h goes into the latch but is not latched yet.
Out	**FC00h + 3Fh** an **00h** (inactive CMD). This latches the high address through the disabling D31, ‘closes the door’.
Out	**FC00h + 3Eh DATA** (the data byte to be written). DATA byte goes into the latch but is not latched yet.
Out	**FC00h + 3Fh** a **C0h** (CMD to write the data byte). This latches the data byte through changing the decode bits and begins to write NVRAM data operation.
In	**FC00h + 3Fh** until **D31 = 0** (not busy).
Out	**FC00h + 3Fh** an **E0h** (CMD to read the address latched).
In	**FC00h + 3Fh** until **D31 = 0** (not busy).
In	**FC00h + 3Eh** the **data.**

This example will read 1 byte from NVRAM location 0040h:

In	**FC00h + 3Fh** (offset of NVRAM Access Control Register) until **D31 = 0** (not busy).
Out	**FC00h + 3Fh** an **80h** (CMD to load the low address byte). This sets decode bits and opens door for low address latch.
Out	**FC00h + 3Eh** (offset of Address/Data Register) **40h** (the low byte of the address desired) 40h goes into latch but is not latched yet.
Out	**FC00h + 3Fh** an **A0h** (CMD to load the high address byte). This latches the low address through changing the decode bits and opens the door for the high address latch.
Out	**FC00h + 3Eh** a **00h** (the high byte of the address desired) 00h goes into latch but is not latched yet.
Out	**FC00h + 3Fh** an **E0h** (CMD to read NVRAM data). This latches the high address through changing the decode bits and begins to read the NVRAM data operation.
In	**FC00h + 3Fh** until **D31 = 0** (not busy).
In	**FC00h + 3Eh** the **data.**

This example will read 1 byte from NVRAM location 0041h and contains an extra step to demonstrate D31 operation:

In	**FC00h + 3Fh** (offset of NVRAM Access Control Register) until **D31 = 0** (not busy).
Out	**FC00h + 3Fh** an **80h** (CMD to load the low address byte). This sets decode bits and opens the door for low address latch.
Out	**FC00h + 3Eh** (offset of Address/Data Register) **40h** (the low byte of the address desired) 40h goes into latch but is not latched yet.
Out	**FC00h + 3Eh** (offset of Address/Data Register) **41h** (the low byte of the address desired) 41h goes into latch but is not latched yet.
Out	**FC00h + 3Fh** an **A0h** (CMD to load the high address byte). This latches the low address through changing the decode bits and opens the door for the high address latch.
Out	**FC00h + 3Eh 00h** (the high byte of the address desired) 00h goes into latch but is not latched yet.
Out	**FC00h + 3Fh** an **E0h** (CMD to read the address latched). This latches the high address through changing the decode bits and begins the read NVRAM data operation.
In	**FC00h + 3Fh** until **D31 = 0** (not busy).
In	**FC00h + 3Eh** the **data.**

Notes:

1. Latched addresses do not automatically increment after a read or write. They must be loaded with new values.
2. Latched addresses remain after reads and writes. It is allowable to only update one address byte for the next access.
3. A processor may perform a one word write to load an address byte and control command simultaneously.

nv Memory Device Timing Requirements

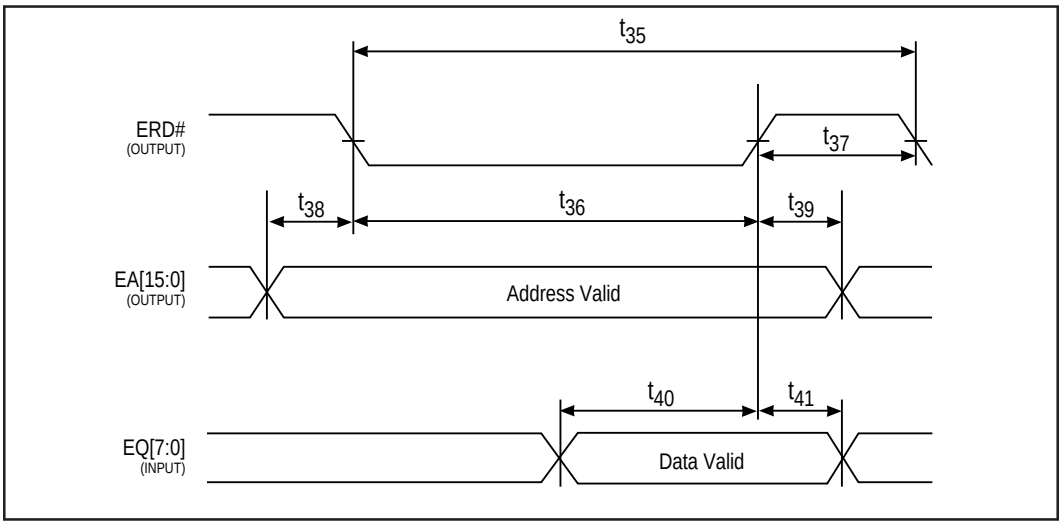
For serial nv memory devices, the serial clock output frequency is the PCI clock frequency divided by 512. This is approximately 65 KHz (with a 33 MHz PCI clock). Any serial memory device that operates at this frequency is compatible with the S5933.

For byte-wide accesses, the S5933 generates the waveforms shown in Figures 5 and 6. Figure 5 shows an nv memory read operation. Figure 6 shows an nv memory write operation. Read operations are always the same length. Write operations, due to the characteristics of reprogrammable nv memory devices, may be controlled through a programming sequence.

Memory Device Requirements for Read Accesses

Timing	Spec.	T = 30 ns
Read cycle time	8T(max)	240 ns
Address valid to data valid	7T–10(max)	200 ns
Address valid to read active	T(max)	30 ns
Read active to data valid	6T–10(max)	170 ns
Read pulse width	6T(max)	180 ns
Data hold from read inactive	—	2 ns

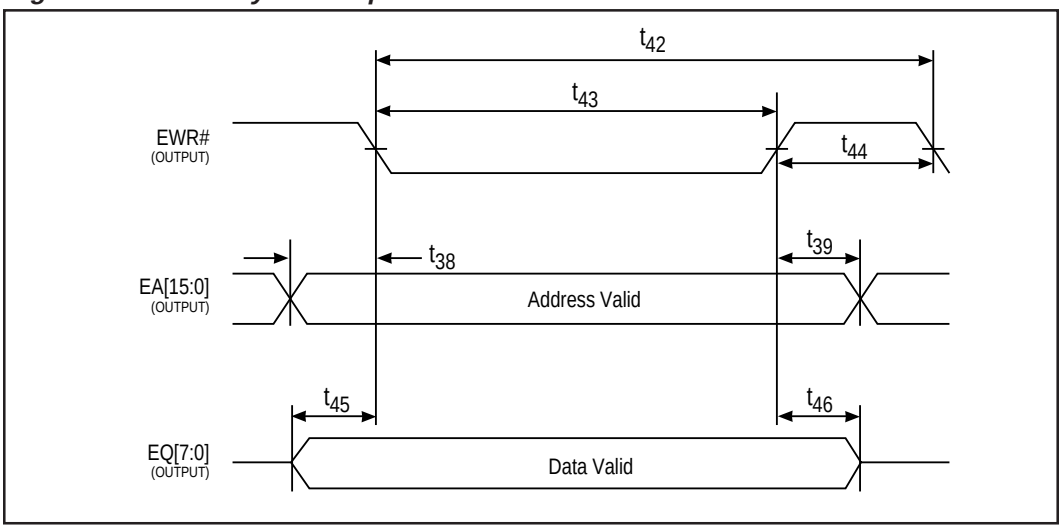
Figure 5. nv Memory Read Operation



Memory Device Requirements for Write Accesses

Timing	Spec.	T = 30 ns
Write cycle time	8T	Note 1
Address valid to write active	T(max)	30 ns
Data valid to write inactive	6T+10(max)	190 ns
Data hold from write inactive	T(max)	30 ns
Write pulse width	6T(max)	180 ns
Write inactive	Note 2	2 ns

Figure 6. nv Memory Write Operation



MAILBOX OVERVIEW

The S5933 has eight 32-bit mailbox registers. The mailboxes are useful for passing command and status information between the Add-On and the PCI bus. The PCI interface has four incoming mailboxes (Add-On to PCI) and four outgoing mailboxes (PCI to Add-On). The Add-On interface has four incoming mailboxes (PCI to Add-On) and four outgoing mailboxes (Add-On to PCI). The PCI incoming and Add-On outgoing mailboxes are the same, internally. The Add-On incoming and PCI outgoing mailboxes are also the same, internally.

The mailbox status may be monitored in two ways. The PCI and Add-On interfaces each have a mailbox status register to indicate the empty/full status of bytes within the mailboxes. The mailboxes may also be configured to generate interrupts to the PCI and/or Add-On interface. One outgoing and one incoming mailbox on each interface can be configured to generate interrupts.

FUNCTIONAL DESCRIPTION

Figure 1 shows a block diagram of the PCI to Add-On mailbox registers. Add-On incoming mailbox read accesses pass through an output interlock latch. This prevents a PCI bus write to a PCI outgoing mailbox from corrupting data being read by the Add-On. Figure 2 shows a block diagram of the Add-On to PCI mailbox registers. PCI incoming mailbox reads also pass through an interlocking mechanism. This prevents an Add-On write to an outgoing mailbox from corrupting data being read by the PCI bus. The following sections describe the mailbox flag functionality and the mailbox interrupt capabilities.

Figure 1. Block Diagram - PCI to Add-On Mailbox Register

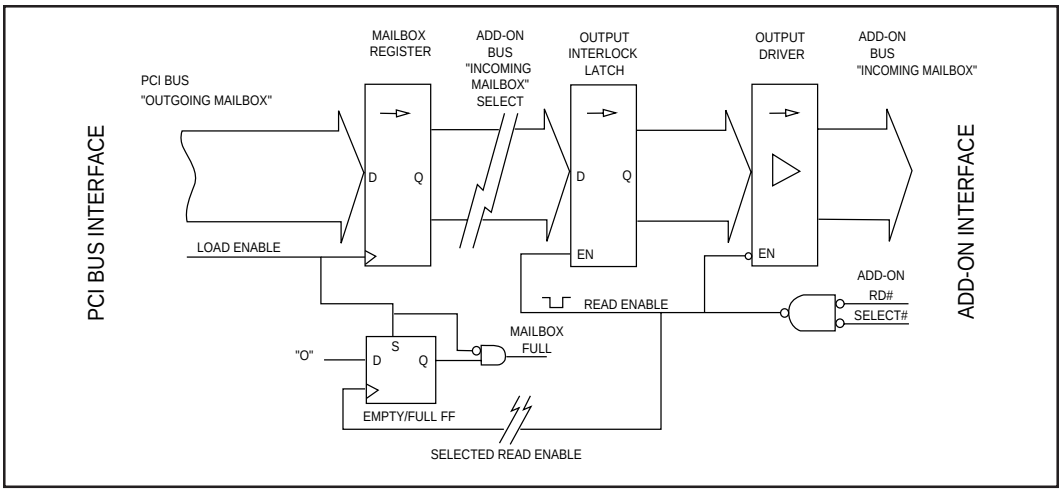
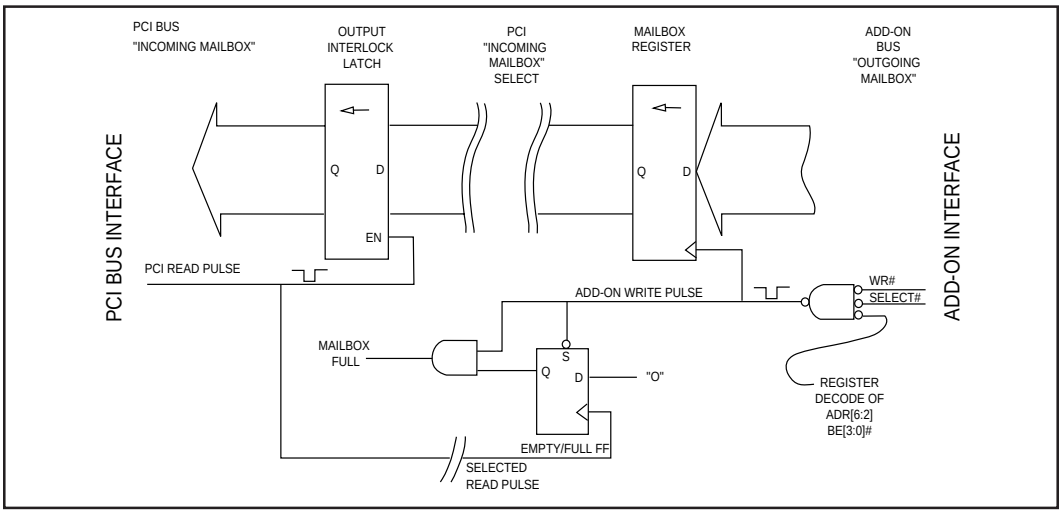


Figure 2. Block Diagram - Add-On to PCI Mailbox Register



Mailbox Empty/Full Conditions

The PCI and Add-On interfaces each have a mailbox status register. The PCI Mailbox Empty/Full Status (MBEF) and Add-On Mailbox Empty/Full Status (AMBEF) Registers indicate the status of all bytes within the mailbox registers. A write to an outgoing mailbox sets the status bits for that mailbox. The byte enables determine which bytes within the mailbox become full (and which status bits are set).

An outgoing mailbox for one interface is an incoming mailbox for the other. Therefore, incoming mailbox status bits on one interface are identical to the corresponding outgoing mailbox status bits on the other interface. The following list shows the relationship between the mailbox registers on the PCI and Add-On interfaces.

PCI Interface		Add-On Interface
Outgoing Mailbox 1	=	Incoming Mailbox 1
Outgoing Mailbox 2	=	Incoming Mailbox 2
Outgoing Mailbox 3	=	Incoming Mailbox 3
Outgoing Mailbox 4	=	Incoming Mailbox 4
Incoming Mailbox 1	=	Outgoing Mailbox 1
Incoming Mailbox 2	=	Outgoing Mailbox 2
Incoming Mailbox 3	=	Outgoing Mailbox 3
Incoming Mailbox 4	=	Outgoing Mailbox 4
PCI Mailbox Empty/Full	=	Add-On Mailbox Empty/Full

A write to an outgoing mailbox also writes data into the incoming mailbox on the other interface. It also sets the status bits for the outgoing mailbox and the status bits for the incoming mailbox on the other interface. Reading the incoming mailbox clears all corresponding status bits in the Add-On and PCI mailbox status registers (AMBEF and MBEF).

For example, a PCI write is performed to the PCI outgoing mailbox 2, writing bytes 0 and 1 (BE0# and BE1# asserted). Reading the PCI Mailbox Empty/Full Status Register (MBEF) indicates that bits 4 and 5 are set. These bits indicate that outgoing mailbox 2, bytes 0 and 1 are full. Reading the Add-On Mailbox Empty/Full Status Register (AMBEF) shows that bits 4 and 5 in this register are also set, indicating Add-On incoming mailbox 2, bytes 0 and 1 are full. An Add-On read of incoming mailbox 2, bytes 0 and 1 clears the status bits in both the MBEF and AMBEF status registers.

To reset individual flags in the MBEF and AMBEF registers, the corresponding byte must be read from the incoming mailbox. The PCI and Add-On mailbox status registers, MBEF and AMBEF, are read-only. Mailbox flags may be globally reset from either the PCI interface or the Add-On interface. The PCI Bus Master Control/Status Register (MCSR) and the Add-On General Control/Status Register (AGCSTS) each have a bit to reset all of the mailbox status flags.

Mailbox Interrupts

The designer has the option to generate interrupts to the PCI and Add-On interfaces when specific mailbox events occur. The PCI and Add-On interfaces can each define two conditions where interrupts may be generated. An interrupt can be generated when an incoming mailbox becomes full and/or when an outgoing mailbox becomes empty. A specific byte within a specific mailbox is selected to generate the interrupt. The conditions defined to generate interrupts to the PCI interface do not have to be the same as the conditions defined for the Add-On interface. Interrupts are cleared through software.

For incoming mailbox interrupts, when the specified byte becomes full, an interrupt is generated. The interrupt might be used to indicate command or status information has been provided, and must be read. For PCI incoming mailbox interrupts, the S5933 asserts the PCI interrupt, INTA#. For Add-On incoming mailbox interrupts, the S5933 asserts the Add-On interrupt, IRQ#.

For outgoing mailbox interrupts, when the specified byte becomes empty, an interrupt is generated. The interrupt might be used to indicate that the other interface has received the last information sent and more may be written. For PCI outgoing mailbox interrupts, the S5933 asserts the PCI interrupt, INTA#. For Add-On outgoing mailbox interrupts, the S5933 asserts the Add-On interrupt, IRQ#.

Add-On Outgoing Mailbox 4, Byte 3 Access

PCI incoming mailbox 4, byte 3 (Add-On outgoing mailbox 4, byte 3) does not function exactly like the other mailbox bytes. When an a serial nv memory boot device or no external boot device is used, the S5933 pins EA7:0 are redefined to provide direct external access to Add-On outgoing mailbox 4, byte 3. EA8 is redefined to provide a load clock which may be used to generate a PCI interrupt. The pins are redefined as follows:

Signal Pin	Add-On Outgoing Mailbox
EA0/EMB0	Mailbox 4, bit 24
EA1/EMB1	Mailbox 4, bit 25
EA2/EMB2	Mailbox 4, bit 26
EA3/EMB3	Mailbox 4, bit 27
EA4/EMB4	Mailbox 4, bit 28
EA5/EMB5	Mailbox 4, bit 29
EA6/EMB6	Mailbox 4, bit 30
EA7/EMB7	Mailbox 4, bit 31
EA8/EMBCLK	Mailbox 4, byte 3 load clock

If the S5933 is programmed to generate a PCI interrupt (INTA#), on an Add-On write to outgoing mailbox 4, byte 3, a rising edge on EMBCLK generates a PCI interrupt. The bits EMB7:0 can be read by the PCI bus interface by reading the PCI incoming mailbox 4, byte 3. These bits are useful to indicate various conditions which may have caused the interrupt.

When using the S5933 with a byte-wide boot device, the capability to generate PCI interrupts with Add-On hardware does not exist. In this configuration, PCI incoming mailbox 4, byte 3 (Add-On incoming mailbox 4, byte 3) cannot be used to transfer data from the Add-On - it always returns zeros when read from the PCI bus. This mailbox byte is sacrificed to allow the added functionality provided when a byte-wide boot device is not used.

BUS INTERFACE

The mailboxes appear on the Add-On and PCI bus interfaces as eight operation registers. Four are outgoing mailboxes, four are incoming mailboxes. The mailboxes may be used to generate interrupts to each of the interfaces. The following sections describe the Add-On and PCI bus interfaces for the mailbox registers.

PCI Bus Interface

The mailboxes are only accessible with the S5933 as a PCI target. The mailbox operation registers do not support burst accesses by an initiator. A PCI initiator attempting to burst to the mailbox registers causes the S5933 to respond with a target disconnect with data. PCI writes to full outgoing mailboxes overwrite data currently in that the mailbox. PCI reads from empty incoming mailboxes return the data that was previously contained in the mailbox. Neither of these situations cause a target retry or abort.

PCI incoming and outgoing mailbox interrupts are enabled in the Interrupt Control/Status Register (INTCSR). The mailboxes can generate a PCI interrupt (INTA#) under two conditions (individually enabled). For an incoming mailbox full interrupt, INTA# is asserted on the PCI clock rising edge after the Add-On mailbox write completes. For an outgoing mailbox empty interrupt, INTA# is asserted on the PCI clock rising edge after the Add-On mailbox read completes (the rising edge of RD#). INTA# is deasserted on the next PCI clock rising edge after the PCI access to clear the mailbox interrupt completes (TRDY# deasserted).

Add-On Bus Interface

The Add-On mailbox interface behaves similar to the PCI bus interface. Add-On writes to full outgoing mailboxes overwrite data currently in that mailbox. PCI reads from empty incoming mailboxes return the data that was previously contained in the mailbox.

Add-On incoming and outgoing mailbox interrupts are enabled in the Add-On Interrupt Control/Status Register (AINT). The mailboxes can generate the Add-On IRQ# interrupt under two conditions (individually enabled). For an incoming mailbox full interrupt, IRQ# is asserted one PCI clock period after the PCI mailbox write completes (TRDY# deasserted). For an outgoing mailbox empty interrupt, IRQ# is asserted one PCI clock period after the PCI mailbox read completes (TRDY# deasserted). IRQ# is deasserted immediately when the Add-On clears the mailbox interrupt.

When the S5933 is used with a serial nv memory boot device or no external boot device, the device pins EA8:0 are redefined. EA7:0 become EMB7:0 data inputs and EA8 becomes EMBCLK, a load clock. This configuration allows the Add-On to generate PCI interrupts with a low-to-high transition on EMBCLK. The PCI incoming mailbox interrupt must be enabled and set for mailbox 4, byte3 in the PCI Interrupt Control/Status Register (INTCSR). EMBCLK should begin high and be pulsed low, then high to be recognized. The rising edge of EMBCLK generates the interrupt. The rising edge of EMBCLK also latches in the values on EMB7:0. The S5933 interrupt logic must be cleared (INTA# deasserted) through INTCSR before further EMBCLK interrupts are recognized.

8-Bit and 16-Bit Add-On Interfaces

Some Add-On designs may implement an 8-bit or 16-bit bus interface. The mailboxes do not require a 32-bit Add-On interface. For 8-bit interfaces, the 8-bit data bus may be externally connected to all four bytes of the 32-bit Add-On interface (DQ 31:24, 23:16, 15:8, 7:0 are all connected). The Add-On device reading or writing the mailbox registers may access all mailbox bytes by cycling through the Add-On byte enable inputs. A similar solution applies to 16-bit Add-On buses. This solution works for Add-Ons which always use just 8-bit or just 16-bit accesses.

If the MODE pin is high, indicating a 16-bit Add-On interface, the previous solution may be modified for an 8-bit interface. The difference is that ADR1 must be toggled after the first two accesses to steer the S5933 internal data bus to the upper 16-bits of the mailboxes.

CONFIGURATION

The PCI interface and the Add-On interface each have four incoming mailboxes (IMBx or AIBMx) and four outgoing mailboxes (OMBx or AOMBx) along with a single mailbox status register (MBEF or AMBEF). Outgoing mailboxes are read/write, incoming mailboxes and the mailbox status registers are read-only.

The following sections discuss the registers associated with the mailboxes and accesses required for different modes of mailbox operation.

Mailbox Status

Every byte in each mailbox has a status bit in the Mailbox Empty/Full Status Registers (MBEF and AMBEF). Writing a particular byte into an outgoing mailbox sets the corresponding status bit in both the MBEF and AMBEF registers. A read of a 'full' byte in a mailbox clears the status bit. The MBEF and AMBEF are read-only. Status bits cannot cleared by writes to the status registers.

The S5933 allows the mailbox status bits to be reset through software. The Bus Master Control/Status (MCSR) PCI Operation Register and the Add-On General Control/Status (AGCSTS) Add-On Operation Register each have a bit to reset mailbox status. Writing a '1' to Mailbox Flag Reset bit in the MCSR or the AGCSTS register immediately clears all bits in the both the MBEF and AMBEF registers. Writing a '0' has no effect. The Mailbox Flag Reset bit is write-only.

The flag bits should be monitored when transferring data through the mailboxes. Checking the mailbox status before performing an operation prevents data from being lost or corrupted. The following sequences are suggested for PCI mailbox operations using status polling (interrupts disabled):

Reading a PCI Incoming Mailbox:

1) Check Mailbox Status. Read the mailbox status register to determine if any information has been passed from the Add-On interface.

MBEF	Bits 31:16	If a bit is set, valid data is contained in the corresponding mailbox byte.

2) Read Mailbox(es). Read the mailbox bytes which MBEF indicates are full. This automatically resets the status bits in the MBEF and AMBEF registers.

IMBx	Bits 31:0	Mailbox data.

Writing a PCI Outgoing Mailbox:

1) Check Mailbox Status. Read the mailbox status register to determine if information previously written to the mailbox has been read by the Add-On interface. Writes to full mailbox bytes overwrite data currently in the mailbox (if not already read by the Add-On interface). Repeat until the byte(s) to be written are empty.

MBEF	Bits 15:0	If a bit is set, valid data is contained in the corresponding mailbox byte and has not been read by the Add-On.

2) Write Mailbox(es). Write to the outgoing mailbox byte(s).

OMBx	Bits 31:0	Mailbox data.

Mailbox operations for the Add-On interface are functionally identical. The following sequences are suggested for Add-On mailbox operations using status polling (interrupts disabled):

Reading an Add-On Incoming Mailbox:

1) Check Mailbox Status. Read the mailbox status register to determine if any information has been passed from the PCI interface.

AMBEF	Bits 15:0	If a bit is set, valid data is contained in the corresponding mailbox byte.

2) Read Mailbox(es). Read the mailbox bytes which AMBEF indicates are full. This automatically resets the status bits in the AMBEF and MBEF registers.

AIMBx	Bits 31:0	Mailbox data.

Writing an Add-On Outgoing Mailbox:

1) Check Mailbox Status. Read the mailbox status register to determine if information previously written to the mailbox has been read by the PCI interface. Writes to full mailbox bytes overwrite data currently in the mailbox (if not already read by the PCI interface). Repeat until the byte(s) to be written are empty.

AMBEF	Bits 31:16	If a bit is set, valid data is contained in the corresponding mailbox byte and has not been read by the PCI bus.

2) Write Mailbox(es). Write to the outgoing mailbox byte(s).

AOMBx	Bits 31:0	Mailbox data.

Mailbox Interrupts

Although polling status is useful, in some cases, polling requires continuous actions by the processor reading or writing the mailbox. Mailbox interrupt capabilities are provided to avoid much of the processor overhead required by continuously polling status bits.

The Add-On and PCI interface can each generate interrupts on an incoming mailbox condition and/or an outgoing mailbox condition. These can be individually enabled/disabled. A specific byte in one incoming mailbox and one outgoing mailbox is identified to generate the interrupt(s). The tasks required to setup mailbox interrupts are shown below:

Enabling PCI mailbox interrupts:

1) Enable PCI outgoing mailbox interrupts. A specific byte within one of the outgoing mailboxes is identified to assert INTA# when read by the Add-On interface.

INTCSR	Bit 4	Enable outgoing mailbox interrupts
INTCSR	Bits 3:2	Identify mailbox to generate interrupt
INTCSR	Bits 1:0	Identify mailbox byte to generate interrupt

2) Enable PCI incoming mailbox interrupts. A specific byte within one of the incoming mailboxes is identified to assert INTA# when written by the Add-On interface.

INTCSR	Bit 12	Enable incoming mailbox interrupts
INTCSR	Bits 11:10	Identify mailbox to generate interrupt
INTCSR	Bits 9:8	Identify mailbox byte to generate interrupt

Enabling Add-On mailbox interrupts:

1) Enable Add-On outgoing mailbox interrupts. A specific byte within one of the outgoing mailboxes is identified to assert IRQ# when read by the PCI interface.

AINT	Bit 12	Enable outgoing mailbox interrupts
AINT	Bits 11:10	Identify mailbox to generate interrupt
AINT	Bits 9:8	Identify mailbox byte to generate interrupt

2) Enable Add-On incoming mailbox interrupts. A specific byte within one of the incoming mailboxes is identified to assert IRQ# when written by the PCI interface.

AINT	Bit 4	Enable incoming mailbox interrupts
AINT	Bits 3:2	Identify mailbox to generate interrupt
AINT	Bits 1:0	Identify mailbox byte to generate interrupt

With either the Add-On or PCI interface, these two steps can be performed with a single access to the appropriate register. They are shown separately here for clarity.

Once interrupts are enabled, the interrupt service routine must access the mailboxes and clear the interrupt source. A particular application may not require all of the steps shown. For instance, a design may only use incoming mailbox interrupts and not require support for outgoing mailbox interrupts. The interrupt service routine tasks are shown below:

Servicing a PCI mailbox interrupt (INTA#):

1) Identify the interrupt source(s). Multiple interrupt sources are available on the S5933. The interrupt service routine must verify that a mailbox generated the interrupt (and not some other interrupt source).

INTCSR	Bit 16	PCI outgoing mailbox interrupt indicator
INTCSR	Bit 17	PCI incoming mailbox interrupt indicator

2) Check mailbox status. The mailbox status bits indicate which mailbox bytes must be read or written.

MBEF	Bits 31:16	Full PCI incoming mailbox bytes
MBEF	Bits 15:0	Empty PCI outgoing mailbox bytes

3) Access the mailbox. Based on the contents of MBEF, mailboxes are read or written. Reading an incoming mailbox byte clears the corresponding status bit in MBEF.

OMBx	Bits 31:0	PCI outgoing mailboxes
IMBx	Bits 31:0	PCI incoming mailboxes

4) Clear the interrupt source. The PCI INTA# signal is deasserted by clearing the interrupt request. The request is cleared by writing a ‘1’ to the appropriate bit.

INTCSR	Bit 16	Clear PCI outgoing mailbox interrupt
INTCSR	Bit 17	Clear PCI incoming mailbox interrupt

Servicing an Add-On mailbox interrupt (IRQ#):

1) Identify the interrupt source(s). Multiple interrupt sources are available on the S5933. The interrupt service routine must verify that a mailbox generated the interrupt (and not some other interrupt source).

AINT	Bit 16	Add-On incoming mailbox interrupt indicator
AINT	Bit 17	Add-On outgoing mailbox interrupt indicator

2) Check mailbox status. The mailbox status bits indicate which mailbox bytes must be read or written.

AMBEF	Bits 31:16	Empty Add-On outgoing mailbox bytes
AMBEF	Bits 15:0	Full Add-On incoming mailbox bytes

3) Access the mailbox. Based on the contents of AMBEF, mailboxes are read or written. Reading an incoming mailbox byte clears the corresponding status bit in AMBEF.

AIMBx	Bits 31:0	Add-On incoming mailboxes
AOMBx	Bits 31:0	Add-On outgoing mailboxes

4) Clear the interrupt source. The Add-On IRQ# signal is deasserted by clearing the interrupt request. The request is cleared by writing a '1' to the appropriate bit.

AINT	Bit 16	Clear Add-On incoming mailbox interrupt
AINT	Bit 17	Clear Add-On outgoing mailbox interrupt

In both cases, step 3 involves accessing the mailbox. To allow the incoming mailbox interrupt logic to be cleared, the mailbox status bit must also be cleared. Reading an incoming mailbox clears the status bits. Another option for clearing the status bits is to use the Mailbox Flag Reset bit in the MCSR and AGCSTS registers, but this clears all status bits, not just for a single mailbox or mailbox byte. For outgoing mailbox interrupts, the read of a mailbox register is what generated the interrupt; this ensures the status bits are already clear.

FIFO OVERVIEW

The S5933 has two internal FIFOs. One FIFO is for PCI bus to Add-On bus, the other FIFO is for Add-On bus to PCI bus transfers. Each of these has eight 32-bit registers. The FIFOs are both addressed through a single PCI/Add-On Operation Register offset, but which internal FIFO is accessed is determined by whether the access is a read or write.

The FIFO may be either a PCI target or a PCI initiator. As a target, the FIFO allows a PCI bus master to access Add-On data. The FIFO also allows the S5933 to become a PCI initiator. Read and write address registers and transfer count registers allow the S5933 to perform DMA transfers across the PCI bus. The FIFO may act as initiator and a target at different times in the same application.

The FIFO can be configured to support various Add-On bus configurations. FIFO status and control signals allow simple cascading into an external FIFO, the Add-On bus can be 8-, 16-, or 32-bits wide, and data endian conversion is optional to support any type of Add-On CPU. PCI and Add-On interrupt capabilities are available to support bus mastering through the FIFO.

FUNCTIONAL DESCRIPTION

The S5933 FIFO interface allows a high degree of functionality and flexibility. Different FIFO management schemes, endian conversion schemes, and advance conditions allow for a wide variety of Add-On interfaces. Applications may implement the FIFO as either a PCI target or program it to enable the S5933 to be a PCI initiator (bus master). The following sections describe, on a functional level, the capabilities of the S5933 FIFO interface.

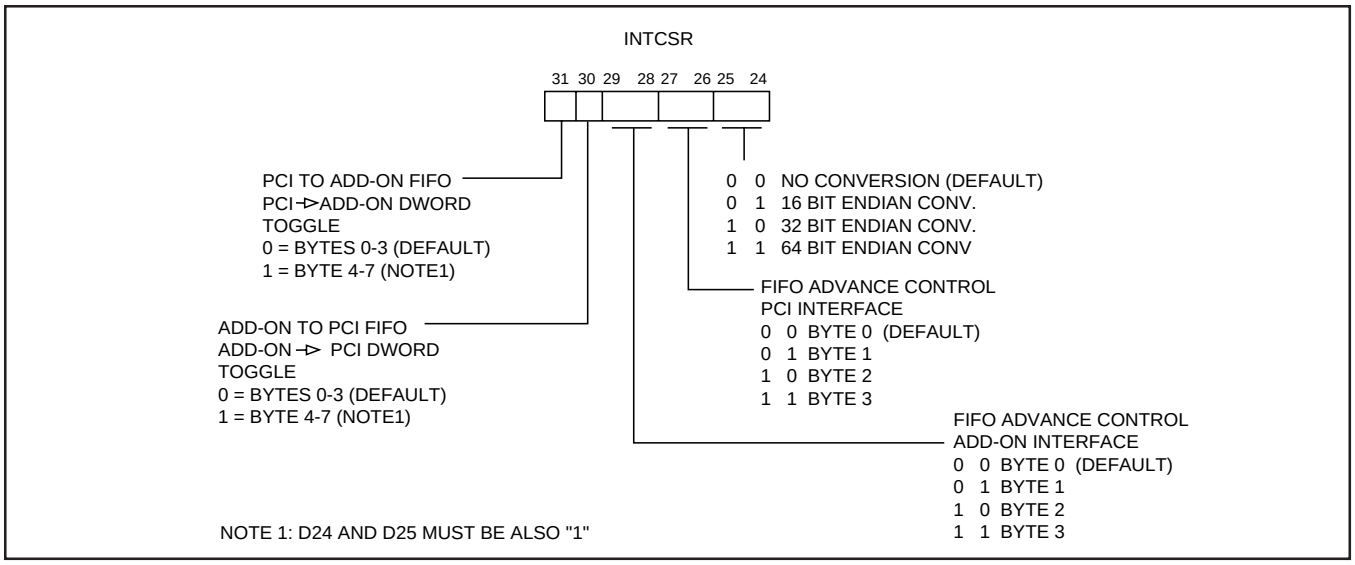
FIFO Buffer Management and Endian Conversion

The S5933 provides a high degree of flexibility for controlling the data flow through the FIFO. Each FIFO (PCI to Add-On and Add-On to PCI) has a specific FIFO advance condition. For FIFO writes, the byte which signifies a location is full is configurable. For FIFO reads, the byte which signifies a location is empty is configurable. This ability is useful for transferring data through the FIFO with Add-Ons which are not 32-bits wide. Endian conversion may also be performed on data passing through the FIFO.

FIFO Advance Conditions

The specific byte lane used to advance the FIFO, when accessed, is determined individually for each FIFO interface (PCI and Add-On). The control bits to set the advance condition are D29:26 of the Interrupt Control/Status Register (INTCSR) in the PCI Operation Registers (Figure 1). The default FIFO advance condition is set to byte 0. With the default setting, a write to the FIFO with BE0# asserted indicates that the FIFO location is now full, advancing the FIFO pointer to the next location. BE0# does not have to be the only byte enable asserted. Note, the FIFO advance condition may be different for the PCI to Add-On FIFO and the Add-On to PCI FIFO directions.

Figure 1. INTCSR FIFO Advance and Endian Control Bits



The configurable FIFO advance condition may be used to transfer data to and from Add-On interfaces which are not 32-bits wide. For a 16-bit Add-On bus, the Add-On to PCI FIFO advance condition can be set to byte 2. This allows a 16-bit write to the lower 16-bits of the FIFO register (bytes 0 and 1) and a second write to the upper 16-bits of the FIFO register (bytes 2 and 3). The FIFO does not advance until the second access. This allows the Add-On to operate with 16-bit data, while the PCI bus maintains a 32-bit data path.

Notes:

1. During operation, the INTCSR FIFO advance condition bits (D29:26) should only be changed when the FIFO is empty and is idle on both the Add-On and PCI interfaces.

Endian Conversion

Bits D31:30 and D25:24 of the INTCSR PCI Operation Register control endian conversion operations for the FIFO (Figure 1). When endian conversion is performed, it affects data passing in either direction through the FIFO interface. Figures 2a and 2b show 16-bit and 32-bit endian conversion. It is important to note that endian conversion is performed on data BEFORE it enters the FIFO. This affects the FIFO advance condition. Example: the FIFO is configured to perform 32-bit endian conversion on data, and the FIFO advance condition is set to byte 0. Byte 3 is written into the FIFO (BE3# asserted). After the endian conversion, byte 3 becomes byte 0, and the FIFO advances. This behavior must be considered when not performing full 32-bit accesses to the FIFO.

Notes:

1. During operation, the INTCSR FIFO endian conversion bits (D25:24) and 64-bit access bits (D31:30) should only be changed when the FIFO is empty and is idle on both the Add-On and PCI interfaces.

Figure 2a. 16-bit Endian Conversion

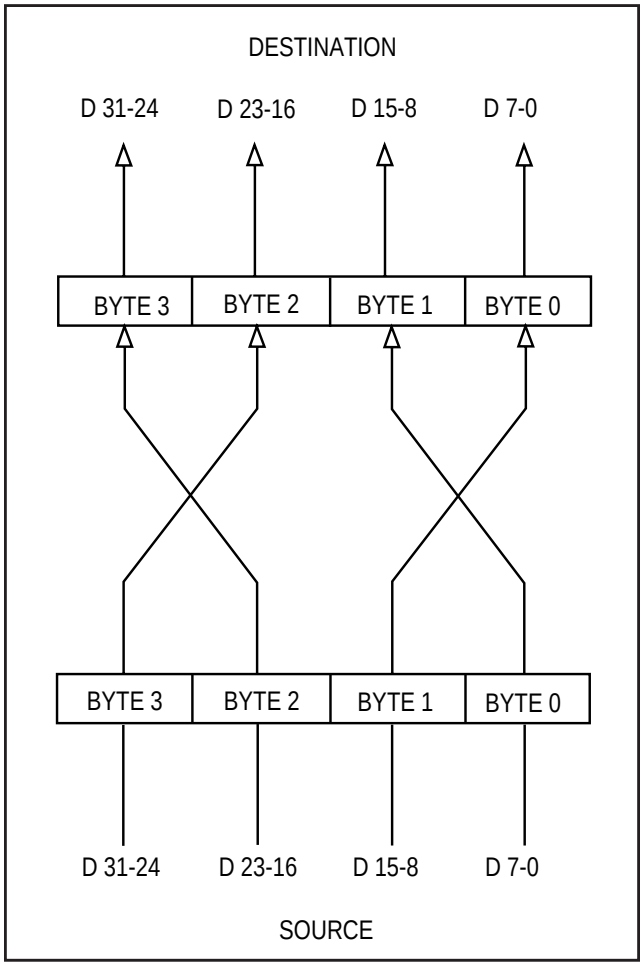
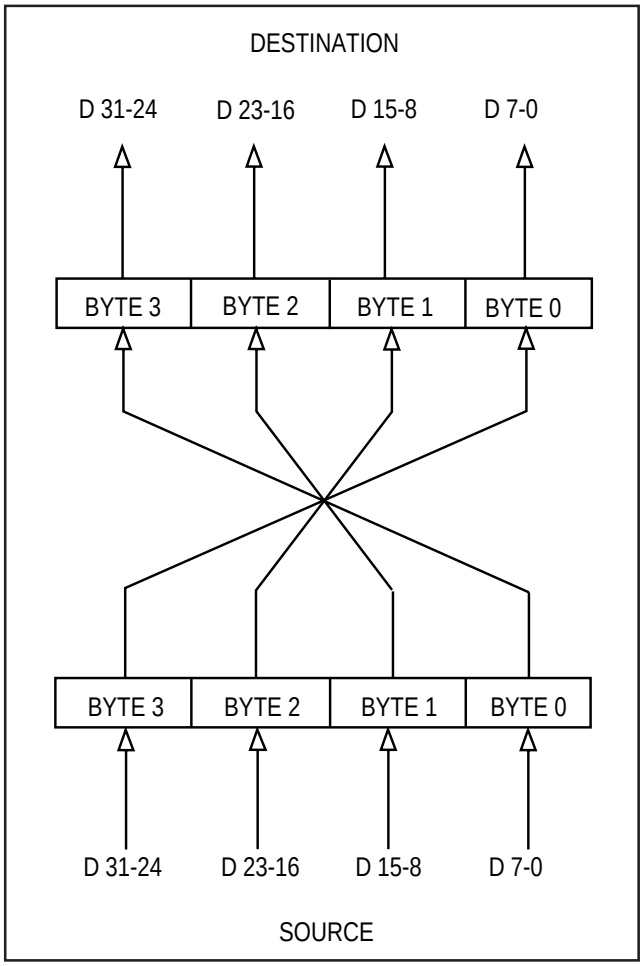


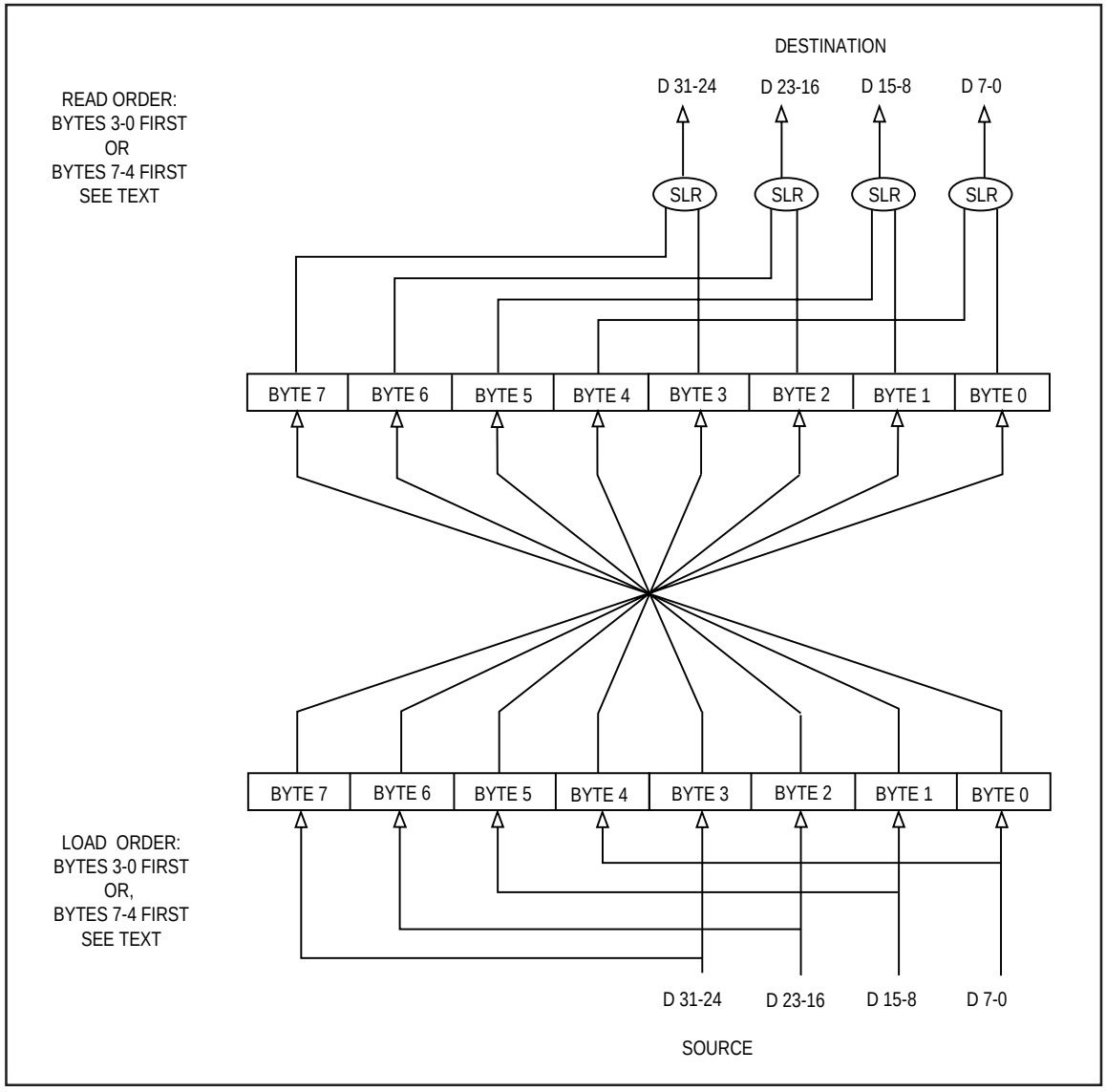
Figure 2b. 32-bit Endian Conversion



64-Bit Endian Conversion

Because the S5933 interfaces to a 32-bit PCI bus, special operation is required to handle 64-bit data endian conversion. Figure 2c shows 64-bit endian conversion. The S5933 must know whether the lower 32-bits enter the FIFO first or the upper 32-bits enter the FIFO first. INTCSR D31:30 identify which method is used by the application. These bits toggle after each 32-bit operation to indicate if half or all of a 64-bit data operation has been completed. The initial state of these bits establishes the loading and emptying order for 64-bit data during operation.

Figure 2c. 64-bit Endian Conversion



Add-On FIFO Status Indicators

The Add-On interface implements FIFO status pins to indicate the full and empty conditions of the PCI to Add-On and Add-On to PCI FIFOs. These may be used by the Add-On to allow data transfers between the FIFO and memory, a peripheral, or even a cascaded external FIFO. The RDEMPTY and WRFULL status outputs are always available to the Add-On. Additional status signals are multiplexed with the byte-wide, non-volatile memory interface pins. If the S5933 is configured for Add-On initiated bus mastering, these status signals also become available to the Add-On. FIFO status is also indicated by bits in the Add-On General Control/Status and Bus Master Control/Status Registers. The table below lists all FIFO status outputs and their functions.

Signal	Function
RDEMPTY	Indicates empty condition of the PCI to Add-On FIFO
WRFULL	Indicates full condition of the Add-On to PCI FIFO
FRF	Indicates full condition of the PCI to Add-On FIFO [1]
FWE	Indicates the empty condition of the Add-On to PCI FIFO [1]

1. These signals are only available when a serial non-volatile memory is used and the device is configured for Add-On initiated bus mastering.

Add-On FIFO Control Signals

The Add-On interface implements FIFO control pins to manipulate the S5933 FIFOs. These may be used by Add-On to control data transfer between the FIFO and memory, a peripheral, or even a cascaded external FIFO. The RDFIFO# and WRFIFO# inputs are always available. These pins allow direct access to the FIFO without generating a standard Add-On register access using RD#, WR#, SELECT#, address pins and the byte enables.

Additional control signals are multiplexed with the byte-wide, non-volatile memory interface pins. If a serial non-volatile memory is used and the S5933 is configured for Add-On initiated bus mastering, these control signals also become available. For PCI initiated bus mastering, AMREN, AMWEN, FRC#, and FWC# functionality is always available through bits in the Bus Master Control/Status and Add-On General Control/Status Registers. The FIFO control inputs are listed below.

Signal	Function
RDFIFO#	Reads data from the PCI to Add-On FIFO
WRFIFO#	Writes data into the Add-On to PCI FIFO
FRC#	Reset PCI to Add-On FIFO pointers and status indicators [1]
FWC#	Reset Add-On to PCI FIFO pointers and status indicators [1]
AMREN	Enable bus mastering for Add-On initiated PCI reads [1]
AMWEN	Enable bus mastering for Add-On initiated PCI writes [1]

1. These signals are only available when a serial non-volatile memory is used and the S5933 is configured for Add-On initiated bus mastering.

PCI Bus Mastering with the FIFO

The S5933 may initiate PCI bus cycles through the FIFO interface. The S5933 allows blocks of data to be transferred to and from the Add-On by specifying a source/destination address on the PCI bus and a transfer byte count. This DMA capability allows data to be transferred across the PCI bus without host CPU intervention.

Initiating a bus master transfer requires programming the appropriate address registers and transfer byte counts. This can be done from either the PCI interface or the Add-On interface. Initiating bus master transfers from the add-on is advantageous because the host CPU does not have to intervene for the S5933 to become a PCI Initiator. At the end of a transfer the S5933 may generate an interrupt to either the PCI bus (for PCI initiated transfers) or Add-On interface (for Add-On initiated transfers).

Add-On Initiated Bus Mastering

If bit 7 in location 45h of an external serial non-volatile memory is zero, the Master Read Address Register (MRAR), Master Write Address Register (MWAR), Master Read Transfer Count (MRTC), and Master Write Transfer Count (MWTC) are accessible only from the Add-On interface. Add-On initiated bus mastering is not possible when a byte-wide boot device is used due to shared device pins. When configured for Add-On initiated bus mastering, the S5933 transfers data until the transfer count reaches zero, or it may be configured to ignore the transfer count.

For bus master transfers initiated by the Add-On interface, some applications may not know the size of the data block to be transferred. To avoid constantly updating the transfer count register, the transfer count may be disabled. Bit 28 in the Add-On General Control/Status Register (AGCSTS) performs this function. Disabling the transfer count also disables the interrupt capabilities. Regardless of whether Add-On transfer count is enabled or disabled, the Add-On Master Read Enable (AMREN) and Add-On Master Write Enable (AMWEN) inputs control when the S5933 asserts or deasserts its request to the PCI bus. When Add-On transfer count is enabled, the S5933 will only request the bus when both the transfer count (read or write) is not zero and the appropriate enable line (AMREN or AMWEN) is active. For Add-On initiated bus mastering, AMWEN and AMREN override the read and write bus mastering enable bits in the Bus Master Control/Status Register (MCSR).

PCI Initiated Bus Mastering

If bit 7 in location 45h of the external non-volatile memory is one, the Master Read Address Register (MRAR), Master Write Address Register (MWAR), Master Read Transfer Count (MRTC), and Master Write Transfer Count (MWTC) are accessible only from the PCI bus interface. In this configuration, the S5933 transfers data until the transfer count reaches zero. The transfer count cannot be disabled for PCI initiated bus mastering. If no external nv memory boot device is used, the S5933 defaults to PCI initiated bus mastering.

Address and Transfer Count Registers

The S5933 has two sets of registers used for bus master transfers. There are two operation registers for bus master read operations and two operation registers for bus master write operations. One operation register is for the transfer address (MWAR and MRAR). The other operation register is for the transfer byte count (MWTC and MRTC).

The address registers are written with the first address of the transfer before bus mastering is enabled. Once a transfer begins, this register is automatically updated to reflect the address of the current transfer. If a PCI target disconnects from an S5933 initiated cycle, the transfer is retried starting from the current address in the register. If bus grant (GNT#) is removed or bus mastering is disabled (using AMREN or AMWEN), the value in the address register reflects the next address to be accessed. Transfers must begin on DWORD boundaries.

The transfer count registers contain the number of bytes to be transferred. The transfer count may be written before or after bus mastering is enabled. If bus mastering is enabled, no transfer occurs until the transfer count is programmed with a non-zero value. Once a transfer begins, this register is automatically updated to reflect the number of bytes remaining to be transferred. If the transfer count registers are disabled (for Add-On initiated bus mastering), transfers begin as soon as bus mastering is enabled.

Although transfers must begin on DWORD boundaries, transfer counts do not have to be multiples of four bytes. For example, if the write transfer count (MWTC) register is programmed with a value of 10 (decimal), the S5933 performs two DWORD writes and a third write with only BE0# and BE1# asserted.

Bus Mastering FIFO Management Schemes

The S5933 provides flexibility in how the FIFO is managed for bus mastering. The FIFO management scheme determines when the S5933 requests the bus to initiate PCI bus cycles. The management scheme is configurable for the PCI to Add-On and Add-On to PCI FIFO (and may be different for each). Bus mastering must be enabled for the management scheme to apply (via the enable bits or AMREN/AMWEN).

For the PCI to Add-On FIFO, there are two management options. The PCI to Add-On FIFO management option is programmed through the Bus Master Control/Status Register (MCSR). The FIFO can be programmed to request the bus when any DWORD location is empty or only when four or more locations are empty. After the S5933 is granted control of the PCI bus, the management scheme does not apply. The device continues to read as long as there is an open FIFO location. When the PCI to Add-On FIFO is full or bus mastering is disabled, the PCI bus request is removed by the S5933.

For the Add-On to PCI FIFO, there are two management options. The Add-On to PCI FIFO management option is programmed through the Bus Master Control/Status Register (MCSR). The FIFO can be programmed to request the bus when any DWORD location is full or only when four or more locations are full. After the S5933 is granted control of the PCI bus, the management scheme does not apply. The device continues to write as long as there is data in the FIFO. When the Add-On to PCI FIFO is empty or bus mastering is disabled, the PCI bus request is removed by the S5933.

There are two special cases for the Add-On to PCI FIFO management scheme. The first case is when the FIFO is programmed to request the PCI bus only when four or more locations are full, but the transfer count is less than 16 bytes. In this situation, the FIFO ignores the management scheme and finishes transferring the data. The second case is when the S5933 is configured for Add-On initiated bus mastering with transfer counts disabled. In this situation, the FIFO management scheme must be set to request the PCI bus when one or more locations are full. AMREN and AMWEN may be used to implement a specific FIFO management scheme.

FIFO Bus Master Cycle Priority

In many applications, the FIFO is used as a PCI initiator performing both PCI reads and writes. This requires a priority scheme be implemented. What happens if the FIFO condition for initiating a PCI read and a PCI write are both met?

Bits D12 and D8 in the Bus Master Control/Status Register (MCSR) control the read and write cycle priority, respectively. If these bits are both set or both clear, priority alternates, beginning with a read cycle. If the read priority is set and the write priority is clear, read cycles take priority. If the write priority is set and the read priority is clear, write cycles take priority. Priority arbitration is only done when neither FIFO has control of the PCI bus (the PCI to Add-On FIFO would never interrupt an Add-On to PCI FIFO transfer).

FIFO Generated Bus Master Interrupts

Interrupts may be generated under certain conditions from the FIFO. If PCI initiated bus mastering is used, INTA# is generated to the PCI interface. If Add-On initiated bus mastering is used, IRQ# is generated to the Add-On interface. Interrupts may be disabled.

FIFO Interrupts may be generated from one or more of the following during bus mastering: read transfer count reaches zero, write transfer count reaches zero, or an error occurs during bus mastering. Error conditions include a target or master abort on the PCI bus. Interrupts on PCI error conditions are only enabled if one or both of the transfer count interrupts are enabled.

The Add-On Interrupt Control/Status Register (AINT) or the Interrupt Control Status Register (INTCSR) indicates the interrupt source. The interrupt service routine may read these registers to determine what action is required. As mailboxes are also capable of generating interrupts, this must also be considered in the service routine. Interrupts are also cleared through these registers.

BUS INTERFACE

The S5933 FIFO may be accessed from the Add-On interface or the PCI interface. Add-On FIFO control and status signals allow a simple interface to the FIFO with either an Add-On CPU or programmable logic. The following section describes the PCI and Add-On interface behavior and hardware interface.

FIFO PCI Interface (Target Mode)

The S5933 FIFO may act as a standard PCI target. FIFO empty/full status may be determined by the PCI initiator by reading the status bits in the PCI Bus Master Control/Status Register (MCSR).

The FIFO occupies a single 32-bit register location within the PCI Operation Registers. **A PCI initiator may not perform burst accesses on the FIFO**. Each data phase of a burst causes the PCI initiator to increment its address counter (even though only the first address is driven at the beginning of the burst). The initiator keeps track of the current address in case a disconnect occurs. This allows the initiator to continue the burst from where the disconnect occurred. If the S5933 FIFO initiated a disconnect during a PCI burst to the FIFO register, the burst would be resumed at an address other than the FIFO location (because the initiator address counter has incremented). The S5933 always signals a disconnect if a burst to any PCI Operation Register is attempted.

Because the PCI to Add-On FIFO and the Add-On to PCI FIFO occupy a single location within the PCI and Add-On Operation Registers, which FIFO is accessed is determined by whether the access is a read or write. This means that once data is written into the FIFO, the value written cannot be read back.

For PCI reads from the Add-On to PCI FIFO, the S5933 asserts TRDY# and completes the PCI cycle (Figure 3). If the PCI bus attempts to read an empty FIFO, the S5933 immediately issues a disconnect with retry (Figure 4). The Add-On to PCI FIFO status indicators change one PCI clock after a PCI read.

For PCI writes to the PCI to Add-On, the S5933 asserts TRDY# and completes the PCI cycle (Figure 5). If the PCI bus attempts to write a full FIFO, the S5933 immediately issues a disconnect with retry (Figure 6). The PCI to Add-On FIFO status indicators change one PCI clock after a PCI write.

FIFO PCI Interface (Initiator Mode)

The S5933 can act as an initiator on the PCI bus. This allows the device to gain control of the PCI bus to transfer data to or from the FIFO. Internal address and transfer count registers control the number of PCI transfers and the locations of the transfers. The following paragraphs assume the proper registers and bits are programmed to enable bus mastering .

PCI read and write transfers from the S5933 are very similar. The FIFO management scheme determines when the S5933 asserts its PCI bus request (REQ#). When bus grant (GNT#) is returned, the device begins running PCI cycles. Once the S5933 controls the bus, the FIFO management scheme is not important. It only determines when PCI bus control is initially requested. PCI bus reads and writes are always performed as bursts by the S5933, if possible.

Figure 3. PCI Read from a Full S5933 FIFO

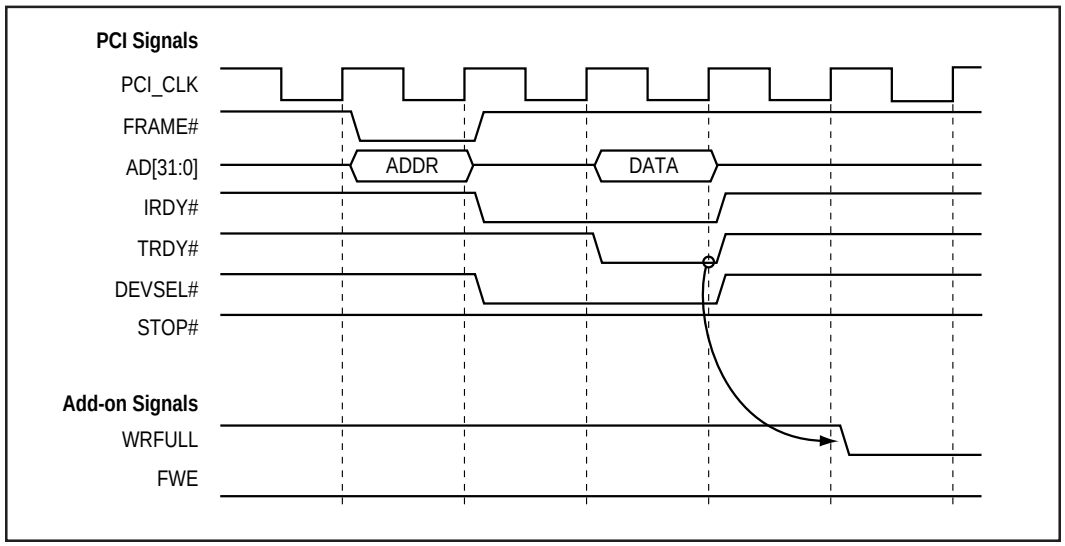


Figure 4. PCI Read from an Empty S5933 FIFO (Target Disconnect)

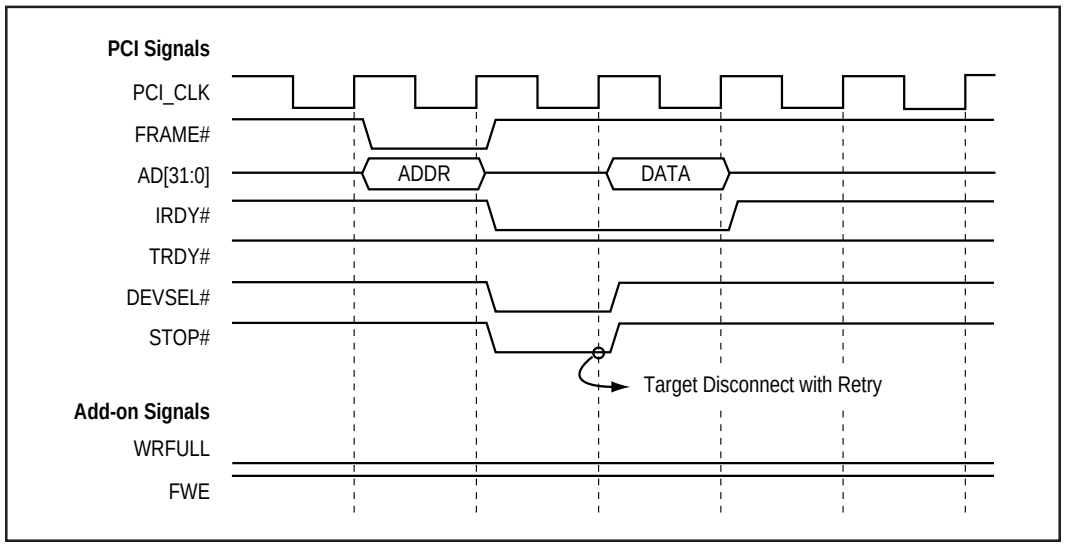


Figure 5. PCI Write to an Empty S5933 FIFO

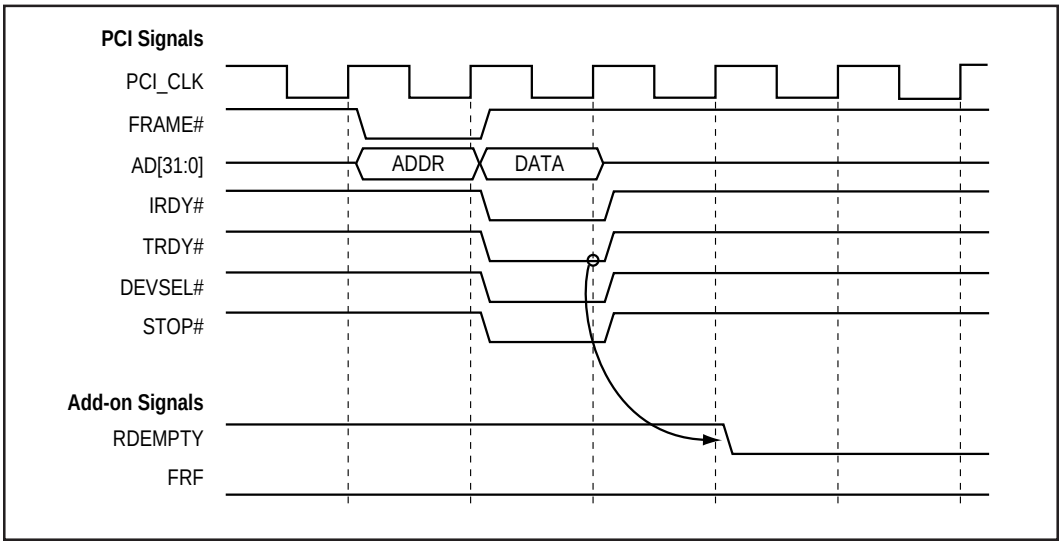
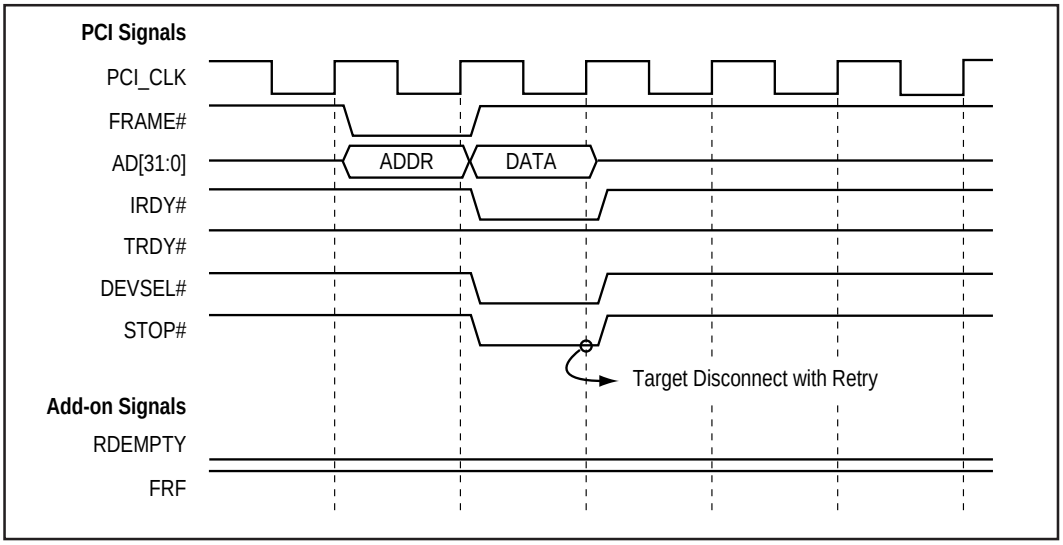


Figure 6. PCI Write to a Full S5933 FIFO (Target Disconnect)



FIFO PCI Bus Master Reads

For PCI read transfers (filling the PCI to Add-On FIFO), read cycles are performed until one of the following occurs:

- Bus Master Read Transfer Count Register (MRTC), if used, reaches zero
- The PCI to Add-On FIFO is full
- GNT# is removed by the PCI bus arbiter
- AMREN is deasserted

If the transfer count is not zero, GNT# remains asserted, and AMREN is asserted, the FIFO continues to read data from the PCI bus until there are no empty locations in the PCI to Add-On FIFO. If the Add-On can empty the FIFO as quickly as it can be filled from the PCI bus, very long bursts are possible. The S5933 deasserts REQ# when it completes the access to fill the last location in the FIFO. Once REQ# is deasserted, it will not be reasserted until the FIFO management condition is met.

FIFO PCI Bus Master Writes

For PCI write transfers (emptying the Add-On to PCI FIFO), write cycles are performed until one of the following occurs:

- Bus Master Write Transfer Count Register (MWTC), if used, reaches zero
- The Add-On to PCI FIFO is empty
- GNT# is removed by the PCI bus arbiter
- AMWEN is deasserted

If the transfer count is not zero, GNT# remains asserted, and AMWEN is asserted, The FIFO continues to write data to the PCI bus until there are is no data in the Add-On to PCI FIFO. If the Add-On can fill the FIFO as quickly as it can be emptied to the PCI bus, very long bursts are possible. The S5933 deasserts REQ# when it completes the access to transfer the last data in the FIFO. Once REQ# is deasserted, it will not be reasserted until the FIFO management condition is met.

Add-On Bus Interface

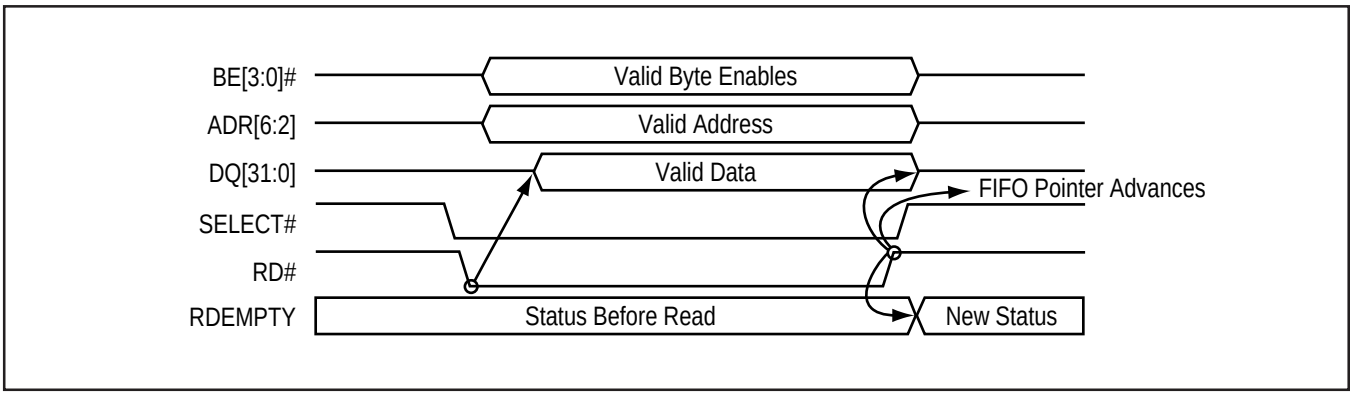
The FIFO register may be accessed in two ways from the Add-On interface. It can be accessed through normal register accesses or directly with the RDFIFO# and WRFIFO# inputs. In addition, the FIFO register can be accessed with synchronous or asynchronous to BPCLK, depending on the S5933 configuration. The Add-On interface also supports datapaths which are not 32-bits. The method used to access the FIFO from the Add-On interface is independent of whether the FIFO is a PCI PCI target or a PCI initiator.

Add-On FIFO Register Accesses

The FIFO may be accessed from the Add-On interface through the Add-On FIFO Port Register (AFIFO) read or write. This is offset 20h in the Add-On Operation Registers. Depending on the device configuration, this register can be accessed either synchronous BPCLK or asynchronous to BPCLK. To access the FIFO as a normal Add-On Operation Register, ADR[6:2], BE[3:0]#, SELECT#, and RD# or WR# are required. The major differences between synchronous and asynchronous modes are when the FIFO pointers advance and the ability to perform burst accesses. The following examples are shown for Add-On FIFO reads.

Figure 7 shows an asynchronous FIFO register read. SELECT# must meet setup and hold times relative to the rising edge of RD#. RD# and SELECT# both asserted enables the DQ outputs, and the first data location in the FIFO is driven onto the bus. The FIFO address and the byte enables must be valid before valid data is driven onto the DQ bus. Data remains valid as long as the address, byte enables, SELECT# and RD# are asserted. Deasserting RD# or SELECT# causes the data bus to float. The rising edge of RD# causes the FIFO pointer to advance. The status outputs are updated to reflect the FIFO condition after it advances.

Figure 7. Asynchronous FIFO Register Read Access Example



When the last location in the PCI to Add-On FIFO is read by the Add-On, the FIFO pointer does not change. If another read is performed before more data enters the FIFO, the previous data is driven. When a write to a full Add-On to PCI FIFO is attempted, nothing happens. No FIFO data is overwritten and the FIFO pointers are not changed. This behavior is the same whether the FIFO is accessed using the direct access inputs or normal Operation Register accesses.

Figure 8 shows a synchronous FIFO register burst access. SELECT# must meet setup and hold times relative to the rising edge of BPCLK. RD# and SELECT# both asserted enables the DQ outputs, and the first data location (data 0) in the FIFO is driven on to the bus. The FIFO address and the byte enables must be valid before valid data is driven onto the DQ bus. Data 0 remains valid until the next rising edge of BPCLK. The rising edge of BPCLK causes the FIFO pointer to advance to the next location (data 1). The next rising edge of BPCLK also advances the FIFO pointer to the next location (data 2). The status outputs reflect the FIFO condition after it advances, and are updated off of the rising edge of BPCLK. When RD# or SELECT# is deasserted, the DQ bus floats. The next time a valid FIFO access occurs and RD# and SELECT# are asserted, data 2 is presented on the DQ bus (as there was no BPCLK edge to advance the FIFO).

Add-On FIFO Direct Access Mode

Instead of generating an address, byte enables, SELECT# and a RD# or WR# strobe for every FIFO access, the S5933 allows a simple, direct access mode. Using RDFIFO# and WRFIFO# is functionally identical to performing a standard AFIFO Port Register access, but requires less logic to implement. Accesses to the FIFO register using the direct access signals are always 32-bits wide. The only exception to this is when the MODE pin is configured for 16-bit operation. In this situation, all accesses are 16-bits wide. The RD# and WR# inputs must be inactive when RDFIFO# or WRFIFO# is active. The ADR[6:2] and BE[3:0]# inputs are ignored.

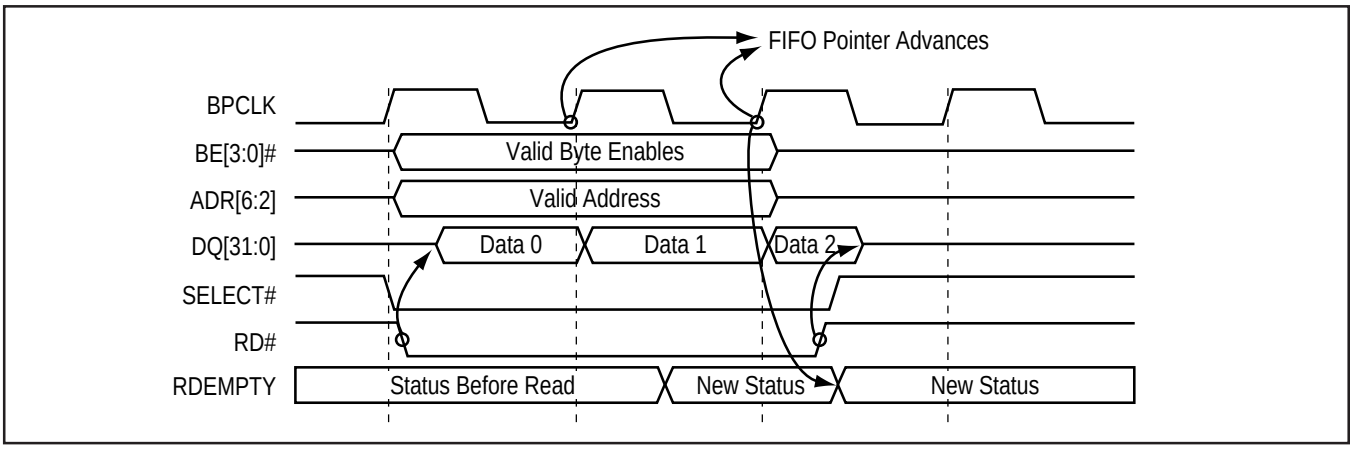
Depending on the device configuration, RDFIFO# and WRFIFO# can act as clocks for data or enables with BPCLK acting as the clock. A Synchronous interface allows higher data rates, and an asynchronous interface is better for slow Add-On logic which may require wait states. The major difference between the synchronous and asynchronous modes is when the FIFO advances.

Figure 9 shows an asynchronous FIFO register direct access using RDFIFO#. The first location in the FIFO is driven onto the bus when RDFIFO# is asserted. Data remains valid as long as RDFIFO# is asserted. The rising edge of RDFIFO# causes the data bus to float and acts as the clock causing the FIFO pointer to advance. The status outputs reflect the FIFO condition after it advances.

Figure 10 shows a synchronous FIFO register direct burst access using RDFIFO#. RDFIFO# acts as an enable and the first data location (data 0) in the FIFO is driven on to the bus when RDFIFO# is asserted. Data 0 remains valid until the next rising edge of BPCLK. The rising edge of BPCLK causes the FIFO pointer to advance to the next location (data 1). The next rising edge of BPCLK advances the FIFO pointer to the next location (data 2). The status outputs reflect the FIFO condition after it advances, and are updated off of the rising edge of BPCLK. When RDFIFO# is deasserted, the DQ bus floats. The next time RDFIFO# is asserted, data 2 is presented on the DQ bus (as there was no BPCLK edge to advance the FIFO).

A synchronous FIFO interface has the advantage of allowing data to be accessed more quickly (in bursts) by the Add-On. As a target, if a full S5933 FIFO is

Figure 8. Synchronous FIFO Register Burst Read Access Example



written (or an empty FIFO is read) by a PCI initiator, the S5933 requests a retry. The faster the Add-On interface can empty (or fill) the FIFO, the less often retries occur. With the S5933 as a PCI initiator, a similar situation occurs. Not emptying or filling the FIFO quickly enough results in the S5933 giving up control of the PCI bus. Higher PCI bus data transfer rates are possible through the FIFO with a synchronous interface.

Additional Status/Control Signals for Add-On Initiated Bus Mastering

If a serial non-volatile memory is used to configure the S5933, and the device is configured for Add-On initiated bus mastering, two additional FIFO status signals and four additional control signals are available to the Add-On interface. The FRF and FWE outputs provide additional FIFO status information. Inputs FRC#, FWC#, AMREN, and AMWEN provide additional FIFO control. Applications may use these signals to monitor/control FIFO flags and PCI bus requests. These new signals are some of the lines that were used for byte-wide nvram interface, but now are reconfigured. The reconfigured lines are as follows:

Outputs:

E_ADDR (15) FRF
FIFO Read Full: Indicates that the PCI to Add-On FIFO is full.

E_ADDR (14) FWE
FIFO Write Empty: Indicates that the Add-On to PCI FIFO is empty.

Inputs:

EQ (7) AMWEN
Add-On bus Mastering Write ENable: This input is driven high to enable bus master writes.

EQ (6) AMREN
Add-On bus Mastering Read ENable: This input is driven high to enable bus master reads.

EQ (5) FRC#
FIFO Read Clear: This line is driven low to clear the PCI to Add-On FIFO.

EQ (4) FWC#
FIFO Write Clear: This line is driven low to clear the Add-On to PCI FIFO.

FRF (PCI to Add-On FIFO full) and FWE (Add-On to PCI FIFO empty) supplement the RDEMPTY and WRFULL status indicators. These additional status outputs provide additional FIFO status information for Add-On FIFO control logic.

Figure 9. Asynchronous FIFO Register RDFIFO# Access Example

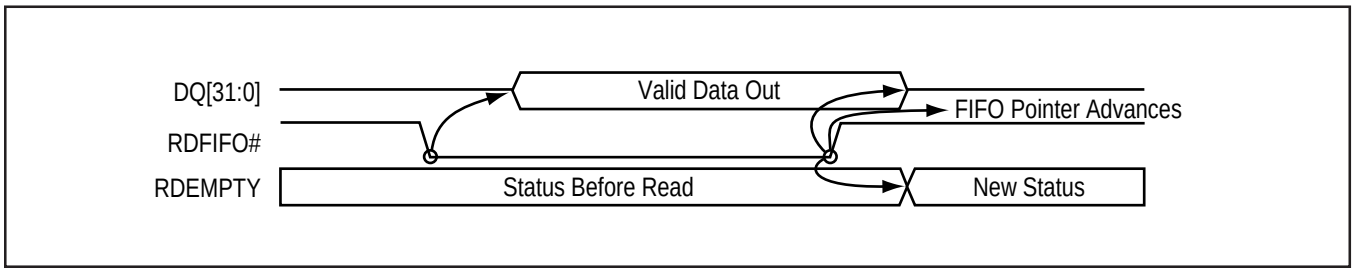
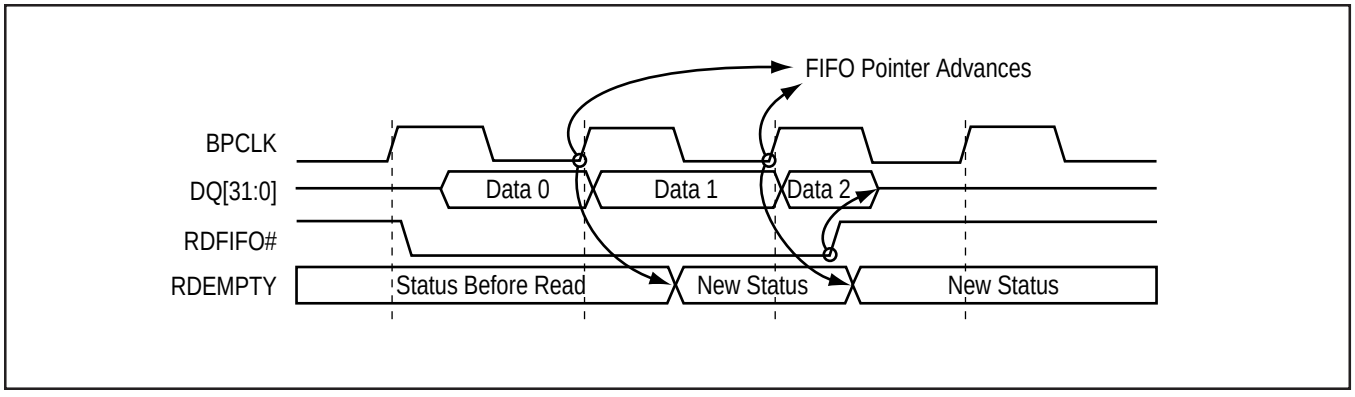


Figure 10. Synchronous FIFO Register Burst RDFIFO# Access Example



The FRC# and FWC# inputs allow Add-On logic to reset the PCI to Add-On or Add-On to PCI FIFO flags. The FIFO flags can always be reset with software through the Add-On General Control/Status Register (AGCSTS) or the Bus Master Control/Status Register (MCSR), but these hardware inputs are useful for designs which do no implement a CPU on the Add-On card. Asserting the FRC# input resets the PCI to Add-On FIFO. Asserting the FWC# input resets the Add-On to PCI FIFO.

The AMREN and AMWEN inputs allow Add-On logic to individually enable and disable bus mastering for the PCI to Add-On and Add-On to PCI FIFO. These inputs override the Bus Master Control/Status Register (MCSR) bus master enable bits. The S5933 may request the PCI bus for the PCI to Add-On FIFO when AMREN is asserted and may request the PCI bus for the Add-On to PCI FIFO when AMWEN is asserted. If AMREN or AMWEN is deasserted, the S5933 removes its PCI bus request and gives up control of the bus.

AMREN and AMWEN are useful for Add-Ons with external FIFOs cascaded into the S5933. For PCI bus master write operations, the entire S5933 Add-On to PCI FIFO and the external FIFO may be filled before enabling bus mastering, providing a single long burst write rather than numerous short bursts.

In some applications, the amount of data to be transferred is not known. During read operations, the S5933, attempting to fill its PCI to Add-On FIFO, may access up to eight memory locations beyond what is required by the Add-On before it stops. In this situation, AMREN can be deasserted to disable PCI reads, and then FRC# can be asserted to flush the unwanted data from the FIFO.

FIFO Generated Add-On Interrupts

For Add-On initiated bus mastering, the S5933 may be configured to generate interrupts to the Add-On interface for the following situations:

- Read transfer count reaches zero
- Write transfer count reaches zero
- An error occurred during the bus master transaction

The interrupt is posted to the Add-On interface with the IRQ# output. A high-to-low transition on this output indicates an interrupt condition. Because there is a single interrupt output and multiple interrupt conditions, the Add-On Interrupt Control/Status Register (AINT) must be read to determine the interrupt source. This register is also used to clear the interrupt, returning IRQ# to its high state. If mailbox interrupts are also used, this must be considered in the interrupt service routine.

8-Bit and 16-Bit FIFO Add-On Interfaces

The S5933 FIFO may also be used to transfer data between the PCI bus and 8-bit or 16-bit Add-On interfaces. This can be done using FIFO advance conditions or the S5933 MODE input pin.

The FIFO may be used as an 8-bit or 16-bit wide FIFO. To use the FIFO as an 8-bit interface, the advance condition should be set for byte 0 (no data is transferred in the upper 3 bytes). To use the FIFO as a 16-bit interface, the advance condition should be set for byte 1 (no data is transferred in the upper 2 bytes). This allows a simple Add-On bus interface, but it has the disadvantage of not efficiently utilizing the PCI bus bandwidth because the host is forced to perform 8-bit or 16-bit accesses to the FIFO on the PCI bus. This is the only way to communicate with an 8-bit Add-On through the FIFO without additional logic to steer byte lanes on the Add-On data bus. Pass-Thru mode is more suited to 8-bit Add-On interfaces.

Implementing a 16-bit wide FIFO is a reasonable solution, but to avoid wasting PCI bus bandwidth, the best method is to allow the PCI bus and the FIFO to operate with 32-bit data. The S5933 can assemble or disassemble 32-bit quantities for the Add-On interface. This is possible through the MODE pin. When MODE is low, the Add-On data bus is 32-bits. When MODE is high, the Add-On data bus is 16-bits. When MODE is configured for 16-bit operation, BE3# becomes ADR1.

With the FIFO direct access signals (RDFIFO# and WRFIFO#), the MODE pin must reflect the actual Add-On data bus width. With MODE = 16-bits, the S5933 automatically takes two consecutive, 16-bit Add-On writes to the FIFO and assembles a 32-bit value. FIFO reads operate in the same manner. Two consecutive Add-On reads empty the 32-bit FIFO register. The 16-bit data bus is internally steered to the lower and upper words of the 32-bit FIFO register.

One consideration needs to be taken when using the FIFO direct access signals and letting the S5933 do byte lane steering internally. The default condition used to advance the FIFO is byte 0. This must be changed to byte 2 or 3. When MODE is configured for a 16-bit Add-On bus, the first 16-bit cycle to the FIFO always accesses the low 16-bits. If the FIFO advance condition is left at byte 0, the FIFO advances after the first 16-bit cycle and the data in the upper 16-bits is directed to the next FIFO location, shifting the data.

Some applications hold the RDFIFO# and WRFIFO# inputs active for a synchronous interface. In 16-bit mode, designs must avoid writing to a full FIFO. The data for the write is lost, but the internal mechanism to direct the 16-bit external data bus to the upper 16-bits of the FIFO register is triggered. This creates a situation where the FIFO is out of step. The next 16-bit FIFO write is directed to the upper 16-bits of the FIFO, and the FIFO advances incorrectly. The WRFULL output should be used to gate the WRFIFO# input to avoid this situation. A similar problem can occur if Add-On logic attempts to read an empty FIFO in 16-bit mode. RDEMPTY should be used to gate the RDFIFO# input to avoid problems with the FIFO getting out of step. In 32-bit mode (MODE = low), these situations do not occur.

If FIFO accesses are done without the direct access signals with MODE configured for 16-bits (using ADR, SELECT#, etc.), external hardware must toggle ADR1 between consecutive 16-bit bus cycles. The FIFO advance condition must be set to correspond to the order the application accesses the upper and lower words in the FIFO register.

CONFIGURATION

The FIFO configuration takes place during initialization and during operation. During initialization, the FIFO hardware interface method and bus master register access rights are defined. During operation, FIFO advance conditions, endian conversion, and bus mastering capabilities are defined. The following section describes the bits and registers which are involved with controlling and monitoring FIFO operation.

FIFO Setup During Initialization

Location 45h in an external non-volatile memory may be used to configure the S5933 FIFO during initialization. If no external non-volatile memory is used, the S5933 defaults to PCI initiated bus master transfers with asynchronous operation for FIFO accesses.

The value of bit 7 in location 45h determines if the address and transfer count registers used in bus mastering are accessible from the PCI bus or from the Add-On bus. Once the configuration information is downloaded from non-volatile memory after reset, the bus mastering initialization method can not be changed. Access to the bus master address and transfer count registers cannot be alternated between the PCI bus and the Add-On interface during operation.

Bits 6 and 5 in location 45h determine if FIFO register accesses using the RDFIFO#, WRFIFO#, RD# and WR# inputs operate asynchronous or synchronous to BPCLK. For asynchronous operation, RDFIFO#, WRFIFO#, RD# and WR# operate as clocks for data. For synchronous operation, RDFIFO#, WRFIFO#, RD# and WR# operate as enables, using BPCLK to clock data. Synchronous operation allows higher data transfer rates.

Location 45h Configuration Bits

Bit 7 Bus Master Register Access

0 Address and transfer count registers only accessible from the Add-On interface

1 Address and transfer count registers only accessible from the PCI interface (default)

Bit 6 RDFIFO#, RD# Operation

0 Synchronous Mode - RDFIFO# and RD# functions as enables

1 Asynchronous Mode - RDFIFO# and RD# functions as clocks (default)

Bit 5 WRFIFO#, WR# Operation

0 Synchronous Mode - WRFIFO# and WR# functions as enables

1 Asynchronous Mode - WRFIFO# and WR# functions as clocks (default)

Bit 0 Target Latency Timer Enable

0 Disable PCI Latency Timer Time Out - Will not disconnect with retry if cannot issue TRDY in specified time

1 Enable PCI Latency Timer Time Out - Will be PCI 2.1 compliant

FIFO Status and Control Bits

The FIFO status can be monitored and the FIFO operation controlled from the PCI Operation Registers and/or the Add-On Operation Registers. The FIFO register resides at offset 20h in the PCI and Add-On Operation Registers.

The Bus Master Control/Status (MCSR) PCI Operation register allows a PCI host to monitor FIFO activity and control FIFO operation. Reset controls allow the PCI to Add-On FIFO and Add-On to PCI FIFO flags to be reset (individually). Status bits indicate if the PCI to Add-On FIFO is empty, has four or more open spaces, or is full. Status bits also indicate if the Add-On to PCI FIFO is empty, has four or more full locations or is full. Finally, FIFO PCI bus mastering is monitored/controlled though this register.

The Add-On General Control/Status (AGCSTS) Add-On Operation Register allows an Add-On CPU to monitor FIFO activity and control FIFO operation. Reset controls allow the PCI to Add-On FIFO and Add-On to PCI FIFO flags to be reset (individually). Status bits indicate if the PCI to Add-On FIFO is empty, has four or more open spaces, or is full. Status bits also indicate if the Add-On to PCI is empty, has four or more full spaces or is full. FIFO bus mastering status may be monitored through this register, but all bus master configuration is through the MCSR PCI Operation Register.

PCI Initiated FIFO Bus Mastering Setup

For PCI initiated bus mastering, the PCI host sets up the S5933 to perform bus master transfers. The following tasks must be completed to setup FIFO bus mastering:

1) Define interrupt capabilities. The PCI to Add-On and/or Add-On to PCI FIFO can generate a PCI interrupt to the host when the transfer count reaches zero.

INTCSR	Bit 15	Enable Interrupt on read transfer count equal zero
INTCSR	Bit 14	Enable Interrupt on write transfer count equal zero

2) Reset FIFO flags. This may not be necessary, but if the state of the FIFO flags is not known, they should be initialized.

MCSR	Bit 26	Reset Add-On to PCI FIFO flags
MCSR	Bit 25	Reset PCI to Add-On FIFO flags

3) Define FIFO management scheme. These bits define what FIFO condition must exist for the PCI bus request (REQ#) to be asserted by the S5933.

MCSR	Bit 13	PCI to Add-On FIFO management scheme
MCSR	Bit 9	Add-On to PCI FIFO management scheme

4) Define PCI to Add-On and Add-On to PCI FIFO priority. These bits determine which FIFO has priority if both meet the defined condition to request the PCI bus. If these bits are the same, priority alternates, with read accesses occurring first.

MCSR	Bit 12	Read vs. write priority
MCSR	Bit 8	Write vs. read priority

5) Define transfer source/destination address. These registers are written with the first address that is to be accessed by the S5933. These address registers are updated after each access to indicate the next address to be accessed. Transfers must start on DWORD boundaries.

MWAR	All	Bus master write address
MRAR	All	Bus master read address

6) Define transfer byte counts. These registers are written with the number of bytes to be transferred. The transfer count does not have to be a multiple of four bytes. These registers are updated after each transfer to reflect the number of bytes remaining to be transferred.

MWTC	All	Write transfer byte count
MRTC	All	Read transfer byte count

7) Enable Bus Mastering. Once steps 1-6 are completed, the FIFO may operate as a PCI bus master. Read and write bus master operation may be independently enabled or disabled.

MCSR	Bit 14	Enable PCI to Add-On FIFO bus mastering
MCSR	Bit 10	Enable Add-On to PCI FIFO bus mastering

The order of the tasks listed above is not particularly important. It is recommended that bus mastering be enabled as the last step. Some applications may choose to leave bus mastering enabled and start transfers by writing a non-zero value to the transfer count registers. This also works, provided the entire transfer count is written in a single access. As a number of the configuration bits and the two enable bits are all in the MCSR register, it may be most efficient for the FIFO configuration bits to be set with the same register access that enables bus mastering.

If interrupts are enabled, a host interrupt service routine is also required. The service routine determines the source of the interrupt and resets the interrupt. As mailbox registers may also be configured to generate interrupts, the exact source of the interrupt is indicated in the PCI Interrupt Control/Status Register (INTCSR). Typically, the interrupt service routine is used to setup the next transfer by writing new addresses and transfer counts, but some applications may also require other actions. If read transfer or write transfer complete interrupts are enabled, master and target abort interrupts are automatically enabled. These indicate a transfer error has occurred. Writing a one to these bits clears the corresponding interrupt.

INTCSR	Bit 21	Target abort caused interrupt
INTCSR	Bit 20	Master abort caused interrupt
INTCSR	Bit 19	Read transfer complete caused interrupt
INTCSR	Bit 18	Write transfer complete caused interrupt

Add-On Initiated FIFO Bus Mastering Setup

For Add-On initiated bus mastering, the Add-On sets up the S5933 to perform bus master transfers. The following tasks must be completed to setup FIFO bus mastering:

1) Define transfer count abilities. For Add-On initiated bus mastering, transfer counts may be either enabled or disabled. Transfer counts for read and write operations cannot be individually enabled.

AGCSTS	Bit 28	Enable transfer count for read and write bus master transfers

2) Define interrupt capabilities. The PCI to Add-On and/or Add-On to PCI FIFO can generate an interrupt to the Add-On when the transfer count reaches zero (if transfer counts are enabled).

AINT	Bit 15	Enable interrupt on read transfer count equal zero
AINT	Bit 14	Enable interrupt on write transfer count equal zero

3) Reset FIFO flags. This may not be necessary, but if the state of the FIFO flags is not known, they should be initialized.

AGCSTS	Bit 25	Reset Add-On to PCI FIFO flags
AGCSTS	Bit 26	Reset PCI to Add-On FIFO flags

4) Define FIFO management scheme. These bits define what FIFO condition must exist for the PCI bus request (REQ#) to be asserted by the S5933. This must be programmed through the PCI interface.

MCSR	Bit 13	PCI to Add-On FIFO management scheme
MCSR	Bit 9	Add-On to PCI FIFO management scheme

5) Define PCI to Add-On and Add-On to PCI FIFO priority. These bits determine which FIFO has priority if both meet the defined condition to request the PCI bus. If these bits are the same, priority alternates, with read accesses occurring first. This must be programmed through the PCI interface.

MCSR	Bit 12	Read vs. write priority
MCSR	Bit 8	Write vs. read priority

6) Define transfer source/destination address. These registers are written with the first address that is to be accessed by the S5933. These address registers are updated after each access to indicate the next address to be accessed. Transfers must start on DWORD boundaries.

MWAR	All	Bus master write address
MRAR	All	Bus master read address

7) Define transfer byte counts. These registers are written with the number of bytes to be transferred. The transfer count does not have to be a multiple of four bytes. These registers are updated after each transfer to reflect the number of bytes remaining to be transferred. If transfer counts are disabled, these registers do not need to be programmed.

MWTC	All	Write transfer byte count
MRTC	All	Read transfer byte count

8) Enable Bus Mastering. Once steps 1-7 are completed, the FIFO may operate as a PCI bus master. Read and write bus master operation may be independently enabled or disabled. The AMREN and AMWEN inputs control bus master enabling for Add-On initiated bus mastering. The MCSR bus master enable bits are ignored for Add-On initiated bus mastering.

It is recommended that bus mastering be enabled as the last step. Some applications may choose to leave bus mastering enabled (AMREN and AMWEN asserted) and start transfers by writing a non-zero value to the transfer count registers (if they are enabled).

If interrupts are enabled, an Add-On CPU interrupt service routine is also required. The service routine determines the source of the interrupt and resets the interrupt. As mailbox registers may also be configured to generate interrupts, the exact source of the interrup is indicated in the Add-On Interrupt Control Register (AINT). Typically, the interrupt service routine is used to setup the next transfer by writing new addresses and transfer counts (if enabled), but some applications may also require other actions. If read transfer or write transfer complete interrupts are enabled, the master/target abort interrupt is automatically enabled. These indicate a transfer error has occurred. Writing a one to these bits clears the corresponding interrupt.

AINT	Bit 21	Master/target abort caused interrupt
AINT	Bit 19	Read transfer complete caused interrupt
AINT	Bit 18	Write transfer complete caused interrupt

PASS-THRU OVERVIEW

The S5933 provides a simple registered access port to the PCI bus. Using a handshaking protocol with Add-On card logic, the PCI bus directly accesses resources on the Add-On. The Pass-Thru data transfer method is very useful for direct Add-On memory access, or accessing registers within peripherals on an Add-On board. Pass-Thru operation requires an external nv memory boot device to define and configure the S5933 Pass-Thru regions.

The S5933 provides four user-configurable Pass-Thru regions. Each region corresponds to a PCI Configuration Base Address Register (BADR1-4). A region represents a block of address space (the block size is user-defined). Each block can be mapped into memory or I/O space. Memory mapped regions can request to be located below 1 MByte (Real Mode address space for a PC). Each region also has a configurable bus width for the Add-On bus interface. An 8-, 16-, or 32-bit Add-On interface may be selected, for use with a variety of Add-On memory or peripheral devices.

Pass-Thru features can be used only when the S5933 is a PCI target. As a target, the S5933 Pass-Thru mode supports single data transfers as well as burst transfers. When accessed with burst transfers, the S5933 supports data transfers at the full PCI bandwidth. The data transfer rate is only limited by the PCI initiator performing the access and the speed of the Add-On logic.

FUNCTIONAL DESCRIPTION

To provide the PCI bus Add-On with direct access to Add-On resources, the S5933 has an internal Pass-Thru Address Register (APTA), and a Pass-Thru Data Register (APTD). These registers are connected to both the PCI bus interface and the Add-On bus interface. This allows a PCI initiator to perform Pass-Thru writes (data transferred from the PCI bus to the Add-On bus) or Pass-Thru reads (PCI bus requests data from the Add-On bus). The S5933 Pass-Thru interface supports both single cycle (one data phase) and burst accesses (multiple data phases).

Pass-Thru Transfers

Data transfers between the PCI bus and the Add-On using the Pass-Thru interface are implemented with a handshaking scheme. If the PCI bus writes to an S5933 Pass-Thru region, Add-On logic must read the data from the S5933 and store it on the Add-On. If the PCI bus reads from a Pass-Thru region, Add-On logic must write data to the S5933.

Some applications may require that an address be passed to the Add-On for Pass-Thru accesses. For example, a 4 Kbyte Pass-Thru region on the PCI bus may correspond to a 4 Kbyte block of SRAM on the Add-On card. If a PCI initiator accessed this region, the Add-On would need to know the offset within the memory device to access. The Pass-Thru Address Register (APTA) allows the Add-On to access address information for the current PCI cycle. When the PCI bus performs burst accesses, the APTA register is updated by the S5933 to reflect the address of the current data phase.

For PCI writes to the Add-On, the S5933 transfers the data from the PCI bus into the Pass-Thru Data Register (APTD). The S5933 captures the data from the PCI bus when TRDY# is asserted. The PCI bus then becomes available for other transfers. When the Pass-Thru data register becomes full, the S5933 asserts the Pass-Thru status signals to indicate to the Add-On that data is present. The Add-On logic may read the data register and assert PTRDY# to indicate the current access is complete. Until the current access completes, the S5933 responds to further Pass-Thru accesses with retries.

For PCI reads from the Add-On, the S5933 asserts the Pass-Thru status signals to indicate to the Add-On that data is required. The Add-On logic should write to the Pass-Thru Data Register and assert PTRDY# to complete the access. The S5933 does not assert TRDY# to the PCI bus until PTRDY# is asserted by Add-On logic. If the Add-On cannot provide data quickly enough, the S5933 signals a retry to the PCI bus. This allows the PCI bus to perform other tasks, rather than waiting for a slow target.

Pass-Thru Status/Control Signals

The S5933 Pass-Thru registers are accessed using the standard Add-On register access pins. The Pass-Thru Address Register (APTA) can, optionally, be accessed using a single, direct access input, PTADR#. Pass-Thru cycle status indicators are provided to control Add-On logic based on the type of Pass-Thru access occurring (single cycle, burst, etc.). The following signals are provided for Pass-Thru operation:

Signal	Function
PTATN#	This output indicates a Pass-Thru access is occurring
PTBURST#	This output indicates the Pass-Thru access is a PCI burst access
PTNUM[1:0]	These outputs indicate which Pass-Thru region decoded the PCI address
PTBE[3:0]#	These outputs indicate which data bytes are valid (PCI writes), or requested (PCI reads)
PTWR	This output indicates if the Pass-Thru access is a PCI read or a write
PTADR#	When asserted, this input drives the Pass-Thru Address Register contents onto the Add-On data bus
PTRDY#	When asserted, this input indicates the current Pass-Thru transfer has been completed by the Add-On
BPCLK	Buffered PCI bus clock output (to synchronize Pass-Thru data register accesses)

Pass-Thru Add-On Data Bus Sizing

Many applications require an 8-bit or 16-bit Add-On bus interface. Pass-Thru regions can be configured to support bus widths other than 32-bits. Each Pass-Thru region can be defined, during initialization, as 8, 16-, or 32-bits. All of the regions do not need to be the same. This feature allows a simple interface to 8- and 16-bit Add-On devices.

To support alternate Add-On bus widths, the S5933 performs internal data bus steering. This allows the Add-On interface to assemble and disassemble 32-bit PCI data using multiple Add-On accesses to the Pass-Thru Data Register (APTD). The Add-On byte enable inputs (BE[3:0]#) are used to access the individual bytes or words within APTD.

BUS INTERFACE

The Pass-Thru interface on the S5933 is a PCI target-only function. Pass-Thru operation allows PCI initiators to read or write resources on the Add-On card. A PCI initiator may access the Add-On with single data phase cycles or multiple data phase bursts.

The Add-On interface implements Pass-Thru status and control signals used by logic to complete data transfers initiated by the PCI bus. The Pass-Thru interface is designed to allow Add-On logic to function without knowledge of PCI bus activity. Add-On logic only needs to react to the Pass-Thru status outputs. The S5933 PCI interface independently interacts with the PCI initiator to control data flow between the devices.

The following sections describe the PCI and Add-On bus interfaces. The PCI interface description provides a basic overview of how the S5933 interacts with the PCI bus, and may be useful in system debugging. The Add-On interface description indicates functions required by Add-On logic and details the Pass-Thru handshaking protocol.

PCI Bus Interface

The S5933 decodes all PCI bus cycle addresses. If the address associated with the current cycle is to one of S5933 Pass-Thru regions, DEVSEL# is asserted. If the Pass-Thru logic is currently idle (not busy finishing a previous Pass-Thru operation), the bus cycle type is decoded and the Add-On Pass-Thru status outputs are set to initiate a transfer on the Add-On side. If the Pass-Thru logic is currently busy completing a previous access, the S5933 signals a retry to PCI initiator.

The following sections describe the behavior of the PCI interface for Pass-Thru accesses to the S5933. Single cycle accesses, burst accesses, and target-initiated retries are detailed.

PCI Pass-Thru Single Cycle Accesses

Single cycle transfers are the simplest PCI bus transaction. Single cycle transfers have an address phase and a single data phase. The PCI bus transaction starts when an initiator drives address and command information onto the PCI bus and asserts FRAME#. The initiator always deasserts frame before the last data phase. For single cycle transfers, FRAME# is only asserted during the address phase (indicating the first data phase is also the last).

When the S5933 sees FRAME# asserted, it samples the address and command information to determine if the bus transaction is intended for it. If the address is within one of the defined Pass-Thru regions, the S5933 accepts the transfer (assert DEVSEL#), and stores the PCI address in the Pass-Thru Address Register (APTA).

For Pass-Thru writes, the S5933 responds immediately (asserting TRDY#) and transfers the data from the PCI bus into the Pass-Thru Data Register (APTD). The S5933 then indicates to the Add-On interface that a Pass-Thru write is taking place and waits for Add-On logic to read the APTD register and complete the transfer (assert PTRDY#). Once the S5933 has captured the data from the PCI bus, the transfer is finished from the PCI bus perspective, and the PCI bus becomes available for other transfers.

For Pass-Thru reads, the S5933 indicates to the Add-On interface that a Pass-Thru read is taking place and waits for Add-On logic to write the Pass-Thru Data Register and complete the transfer (assert PTRDY#). The S5933 completes the cycle when data is written into the data register. If the Add-On cannot complete the write quickly enough, the S5933 requests a retry from the initiator. See target-requested disconnect information.

PCI Pass-Thru Burst Accesses

For PCI Pass-Thru burst accesses, the S5933 captures the PCI address and determines if it falls into one of the defined Pass-Thru regions. Accesses that fall into a Pass-Thru region are accepted by asserting DEVSEL#. The S5933 monitors FRAME# and IRDY# on the PCI bus to identify burst accesses. If the PCI initiator is performing a burst access, the Pass-Thru status indicators notify Add-On logic.

For Pass-Thru burst writes, the S5933 responds immediately (asserting TRDY#). The S5933 transfers the first data phase of the burst into the Pass-Thru Data Register (APTD), and stores the PCI address in the Pass-Thru Address Register (APTA). The Add-On interface completes the transfer and asserts PTRDY#. Every time PTRDY# is asserted by the Add-On, the S5933 begins the next data phase. The next data phase is latched into the data register. For burst accesses, APTA is automatically incremented by the S5933 for each data phase.

For Pass-Thru burst reads, the S5933 claims the PCI cycle (asserting DEVSEL#). The request for data is passed on to Add-On logic and the PCI address is stored in the APTA register. The Add-On interface completes the transfer and asserts PTRDY#. The S5933 then drives the requested data on the PCI bus and asserts TRDY# to begin the next data phase. The APTA register is automatically incremented by the S5933 for each data phase.

PCI Retry Conditions

In some applications, Add-On logic may not be able to respond to Pass-Thru accesses quickly. In this situation, the S5933 disconnects from the PCI bus, signaling a retry. This indicates that the initiator should try the access again at a later time. This allows other PCI cycles to be run while the logic on the slow target completes the Pass-Thru access. Ideally, when the initiator retries the access, the target has completed the access and can respond to the initiator.

With many devices, particularly memories, the first access takes longer than subsequent accesses (assuming they are sequential and not random). For this reason, the PCI specification allows 16 clocks to respond to the first data phase of a PCI cycle and 8 clocks for subsequent data phases (in the case of a burst) before a retry must be requested by the S5933.

The S5933 also requests a retry if an initiator attempts to burst past the end of a Pass-Thru region. The S5933 updates the Pass-Thru Address Register (APTA) for each data phase during bursts, and if the updated address is not within the current Pass-Thru region, a retry is requested.

For example, a PCI system may map a 512 byte Pass-Thru memory region to 0DC000h to 0DC1FFh. A PCI initiator attempts a four DWORD burst with a starting address of 0DC1F8h. The first and second data phases complete (filling the DWORDs at 0DC1F8h and 0DC1FCh), but the third data phase causes the S5933 to request a retry. This forces the initiator to present the address 0DC200h on the PCI bus. If this address is part of another S5933 Pass-Thru region, the device accepts the access.

PCI Write Retries

When the S5933 requests a retry for a PCI Pass-Thru write, it indicates that the Add-On is still completing a previous Pass-Thru write access. The Pass-Thru Address and Data Register contents (APTA and APTD) are still required for the previous Pass-Thru operation and cannot be updated by the PCI interface until the access completes (the Add-On asserts PTRDY#).

When the Add-On is busy completing a Pass-Thru write, the S5933 requests an **immediate** retry for all Pass-Thru region accesses, allowing the PCI bus to perform other operations. PCI Operation Registers may be accessed while the Add-On is still completing a Pass-Thru access. Only Pass-Thru region accesses receive retry requests.

PCI Read Retries

When the S5933 requests a retry for a PCI Pass-Thru read, it indicates that the Add-On could not complete the read in the required time. The Pass-Thru data cannot be read by the PCI interface until the Add-On asserts PTRDY#, indicating the access is complete.

If the retry occurs after the Add-On has completed the Pass-Thru operation by writing the appropriate data into the Pass-Thru data register and asserting PTRDY#, the S5933 asserts DEVSEL# and TRDY# to complete the PCI read. If the Add-On still has not completed the Pass-Thru read, the S5933 waits for the required 16 clocks. If the Add-On completes the access during this time, TRDY# is asserted and the access is finished. If the Add-On cannot complete the access within 16 clocks, another retry is requested.

When the Add-On is busy completing a Pass-Thru read, the S5933 requests an **immediate** retry for all Pass-Thru region accesses, except the region currently completing the previous access. This allows the PCI bus to perform other operations. The next access to the Pass-Thru region which initiated the retry must be to the **same address** which caused the retry. Another initiator accessing the same Pass-Thru region causes the S5933 to respond with the original initiator's data (for reads). S5933 PCI Operation Registers may be accessed while the Add-On is still completing a Pass-Thru access. Only other Pass-Thru region accesses receive retry requests.

Add-On Bus Interface

The Pass-Thru address and data registers can be accessed as Add-On operation registers. The interface to the Pass-Thru registers is described in. The Pass-Thru data register is updated on the rising edge of BPCLK. For this reason, all Pass-Thru inputs must be synchronous to BPCLK. In the following sections the Add-On Pass-Thru interface is described for Pass-Thru single cycle accesses, burst accesses, target-requested retries, and when using 8-bit and 16-bit Add-On data buses.

Single Cycle Pass-Thru Writes

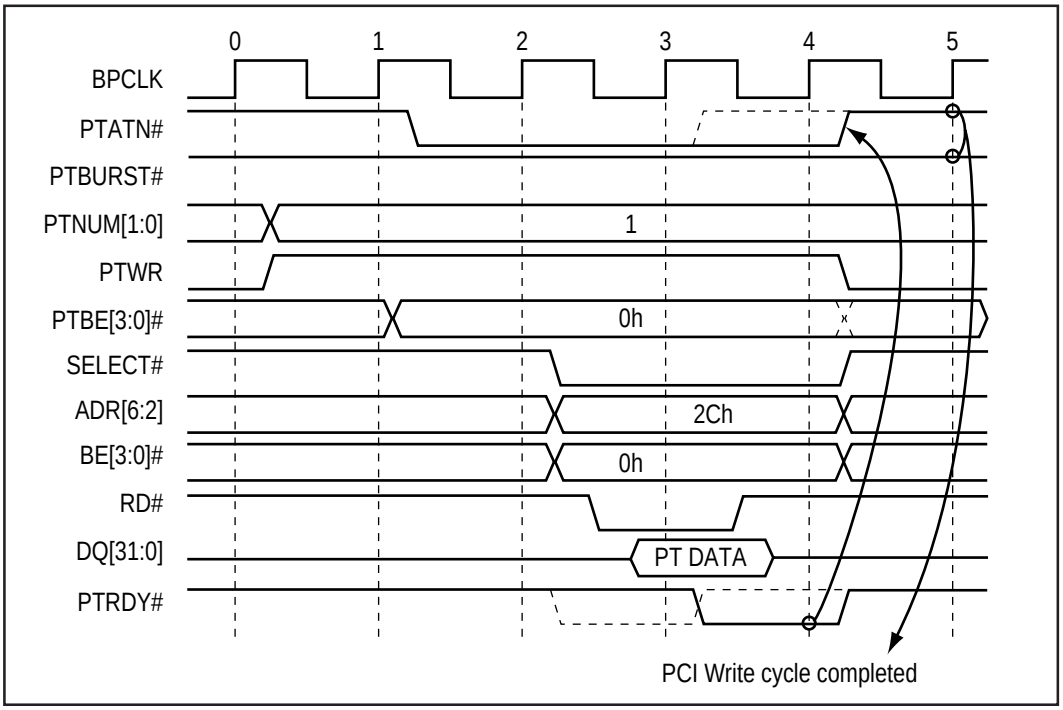
A single cycle Pass-Thru write operation occurs when a PCI initiator writes a single value to a Pass-Thru region. PCI single cycle transfers consists of an address phase and one data phase. During the address phase of the PCI transfer, the S5933 stores the PCI address into the Pass-Thru Address Register (APTA). If the S5933 determines that the address is within one of its defined Pass-Thru regions, it captures the PCI data into the Pass-Thru Data Register (APTD).

Figure 1 shows a single cycle Pass-Thru write access (Add-On read). The Add-On must read the data stored in the APTD register and transfer it to its destination. Note: RD# may be asserted for multiple clocks to allow interfacing with slow Add-On devices. Data remains valid until PTRDY# is asserted.

Note:

For all Add-On accesses using PTADR for address data when in 16 bit mode, ADR[1] must be held low to get the low address word.

Figure 1. Single Cycle Pass-Thru Write



Clock 0: The PCI bus cycle address information is stored in the S5933 Pass-Thru Address Register.

Clock 1: The PCI address is recognized as a write to Pass-Thru region 1. The PCI data is stored in the S5933 Pass-Thru Data Register. PTATN# is asserted to indicate a Pass-Thru access is occurring.

Clock 2: Pass-Thru status signals indicate what action is required by Add-On logic. Pass-Thru status outputs are valid when PTATN# is active and are sampled by the Add-On at the rising edge of clock 2.

PTBURST#	Deasserted. The access has a single data phase.
PTNUM[1:0]	01. Indicates the PCI access is to Pass-Thru region 1.
PTWR	Asserted. The Pass-Thru access is a write.
PTBE[3:0]#	0h. Indicates the Pass-Thru access is 32-bits.

SELECT#, address and byte enable inputs are driven to read the Pass-Thru Data Register at offset 2Ch. DQ[31:0] are driven after RD# and SELECT# are asserted.

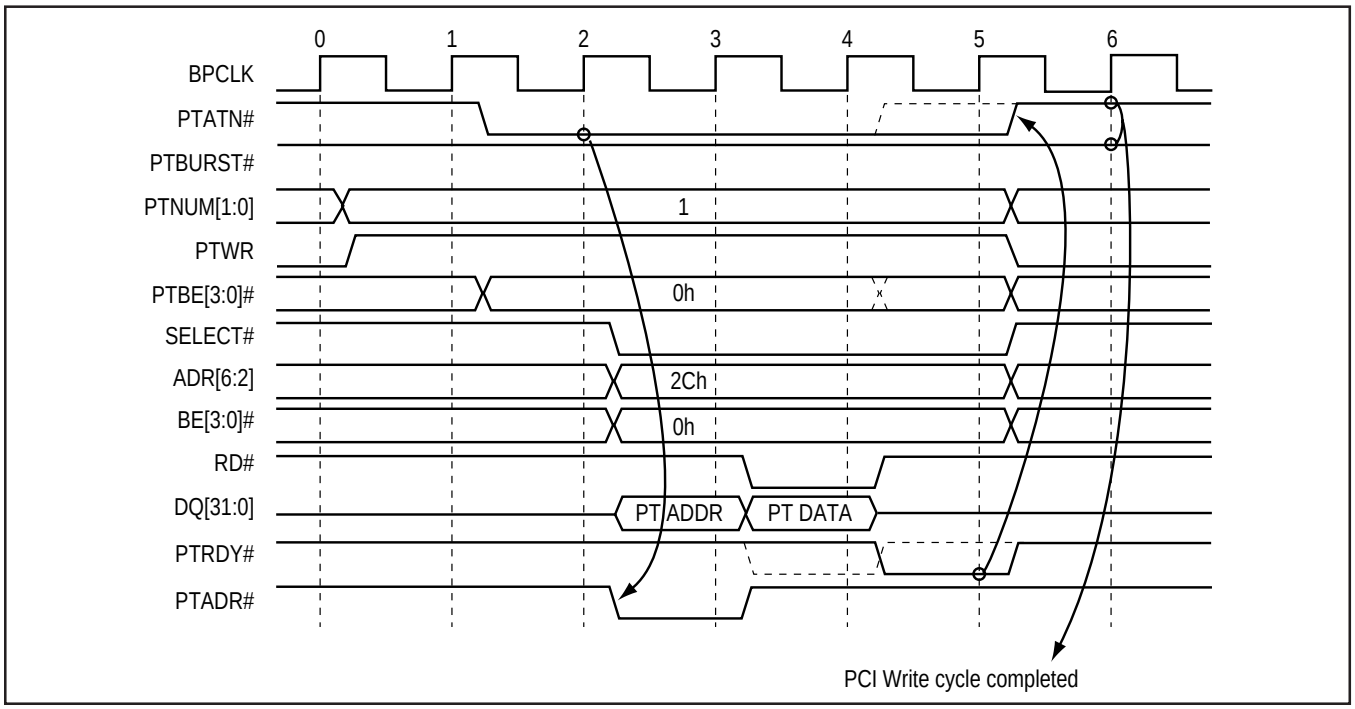
Clock 3: If PTRDY# is asserted at the rising edge of clock 3, PTATN# is immediately deasserted and the Pass-Thru access is completed at clock 4.

Clock 4: If Add-On logic requires more time to read the Pass-Thru Data Register (slower memory or peripherals), PTRDY# can be delayed, extending the cycle. With PTRDY# asserted at the rising edge of clock 4, PTATN# is deasserted and the Pass-Thru access is completed at clock 5.

Clock 5: PTATN# and PTBURST# deasserted at the rising edge of clock 5 indicates the Pass-Thru access is complete. The S5933 can accept new Pass-Thru accesses from the PCI bus at clock 6.

Figure 2 shows a single cycle Pass-Thru write using the Pass-Thru address information. This provides PCI cycle address information to select a specific address location within an Add-On memory or peripheral. Add-On logic must latch the address for use during the data transfer. Typically, the entire 32-bit address is not required. The Add-On may implement a scheme where only the required number of address bits are latched. It may also be useful to use the Pass-Thru region identifiers, PTNUM[1:0] as address lines. For example, Pass-Thru region 1 might be a 64K block of SRAM for data, while Pass-Thru region 2 might be 64K of SRAM for code storage (downloaded from the host during initialization). Using PTNUM0 as address line A16 allows two unique Add-On memory regions to be defined.

Figure 2. Single Cycle Pass-Thru Write with PTADR#



The Add-On PTADR# input directly accesses the Pass-Thru Address Register and drives the contents onto the data bus (no BPCLK rising edge is required). The byte enables, address, and SELECT# inputs are ignored when PTADR# is asserted. RD# and WR# must not be asserted when PTADR# is asserted.

Clock 0: The PCI bus cycle address is stored in the S5933 Pass-Thru Address Register.

Clock 1: The PCI address is recognized as an access to Pass-Thru region 1. PCI data is stored in the S5933 Pass-Thru Data Register. PTATN# is asserted to indicate a Pass-Thru access is occurring.

Clock 2: Pass-Thru status signals indicate what action is required by Add-On logic. Pass-Thru status outputs are valid when PTATN# is active and are sampled by the Add-On at the rising edge of clock 2.

PTBURST#	Deasserted. The access has a single data phase.
PTNUM[1:0]	01. Indicates the PCI access is to Pass-Thru region 1.
PTWR	Asserted. The Pass-Thru access is a write.
PTBE[3:0]#	0h. Indicate the Pass-Thru access is 32-bits.

The PTADR# input is asserted to read the Pass-Thru Address Register. The byte enable, address, and SELECT# inputs are changed during this clock to select the Pass-Thru Data Register during clock cycle 3.

Clock 3: SELECT#, byte enable, and the address inputs remain valid to read the Pass-Thru Data Register at offset 2Ch. RD# is asserted to drive data register contents onto the DQ bus.

Clock 4: If PTRDY# is asserted at the rising edge of clock 4, PTATN# is immediately deasserted and the Pass-Thru access is completed at clock 5.

Clock 5: If Add-On logic requires more time to read the Pass-Thru Data Register (slower memory or peripherals), PTRDY# can be delayed, extending the cycle. PTRDY# asserted at the rising edge of clock 5 causes PTATN# to be immediately deasserted.

Clock 6: PTATN# and PTBURST# deasserted at the rising edge of clock 6 indicates the Pass-Thru access is complete. The S5933 can accept new Pass-Thru accesses from the PCI bus at clock 7.

Single Cycle Pass-Thru Reads

A single cycle Pass-Thru read operation occurs when a PCI initiator reads a single value from a Pass-Thru region. PCI single cycle transfers consists of an address phase and a one data phase. During the address phase of the PCI transfer, the S5933 stores the PCI address into the Pass-Thru Address Register (APTA). If the S5933 determines that the address is within one of its defined Pass-Thru regions, it indicates to the Add-On that a write to the Pass-Thru Data Register (APTD) is required.

Figure 3 shows a single cycle Pass-Thru read access (Add-On write) using PTADR#. The Add-On reads data from a source on the Add-On and writes it to the APTD register.

Clock 0: PCI address information is stored in the S5933 Pass-Thru Address Register. The PCI cycle is recognized as an access to Pass-Thru region 1. PTATN# is asserted by the S5933 to indicate a Pass-Thru access is occurring.

Clock 1: Pass-Thru status signals indicate what action is required by Add-On logic. Pass-Thru status outputs are valid when PTATN# is active and are sampled by the Add-On at the rising edge of clock 1.

PTBURST#	Deasserted. The access has a single data phase.
PTNUM[1:0]	01. Indicates the PCI access was to Pass-Thru region 1.
PTWR	Deasserted. The Pass-Thru access is a read.
PTBE[3:0]#	0h. Indicate the Pass-Thru access is 32-bits.

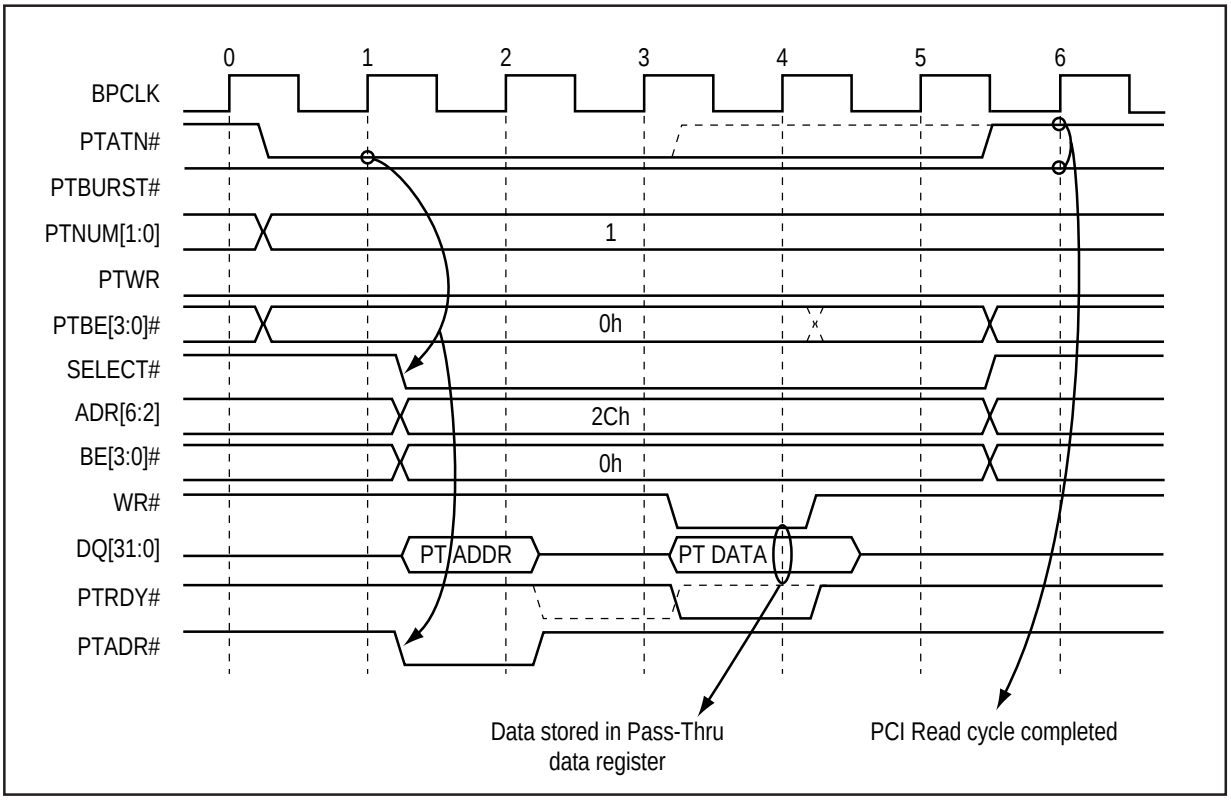
The PTADR# input is asserted to read the Pass-Thru Address Register. The byte enable, address, and SELECT# inputs are changed during this clock to select the Pass-Thru Data Register during clock cycle 3.

Clock 2: This clock is required to avoid contention on the DQ bus. Time must be allowed after PTADR# is deasserted for the DQ outputs to float before Add-On logic attempts to write to the Pass-Thru Data Register.

Clock 3: SELECT#, byte enables, and the address inputs remain valid to write the Pass-Thru Data Register at offset 2Ch. If WR# is asserted at the rising edge of clock 3, data on the DQ bus is latched into APTD.

If PTRDY# is asserted at the rising edge of clock 3, PTATN# is immediately deasserted and the Pass-Thru access is completed at clock 4.

Figure 3. Single Cycle Pass-Thru Read with PTADR#



Clock 4: If Add-On logic requires more time to write the Pass-Thru data register (slower memory or peripherals), PTRDY# can be delayed, extending the cycle. PTRDY# asserted at the rising edge of clock 4 causes PTATN# to be immediately deasserted and the Pass-Thru access is completed at clock 5.

Clock 5: PTATN# and PTBURST# deasserted at the rising edge of clock 5 indicates the Pass-Thru access is complete. The S5933 can accept new Pass-Thru accesses from the PCI bus at clock 6.

Pass-Thru Burst Writes

A Pass-Thru burst write operation occurs when a PCI initiator writes multiple values to a Pass-Thru region. A PCI burst cycle consists of an address phase and multiple data phases. During the address phase of the PCI transfer, the S5933 stores the PCI address into the Pass-Thru Address Register (APTA). If the S5933 determines that the address is within one of its defined Pass-Thru regions, it captures the PCI data into the Pass-Thru Data Register (APTD). After the Add-On completes each read from the Pass-Thru data register (asserts PTRDY#), the next data phase is initiated.

Figure 4 shows a 6 data phase Pass-Thru burst write (Add-On read). In this case, the Add-On asserts PTADR# and then reads multiple data phases from the S5933. This works well for Add-On logic which supports burst cycles. If the Add-On logic does not support burst accesses, PTADR# may be pulsed before each data phase. The S5933 automatically increments the address in the APTA register during PCI burst cycles. In this example PTRDY# is always asserted, indicating Add-On logic is capable of accepting data at a rate of one DWORD per clock cycle.

Clock 0: PCI address information is stored in the S5933 Pass-Thru Address Register.

Clock 1: The PCI address is recognized as an access to Pass-Thru region 1. PCI data for the first data phase is stored in the S5933 Pass-Thru Data Register. PTATN# is asserted by the S5933 to indicate a Pass-Thru access is occurring.

Clock 2: Pass-Thru status signals indicate what action is required by Add-On logic. Pass-Thru status outputs are valid when PTATN# is active and are sampled by the Add-On at the rising edge of clock 2.

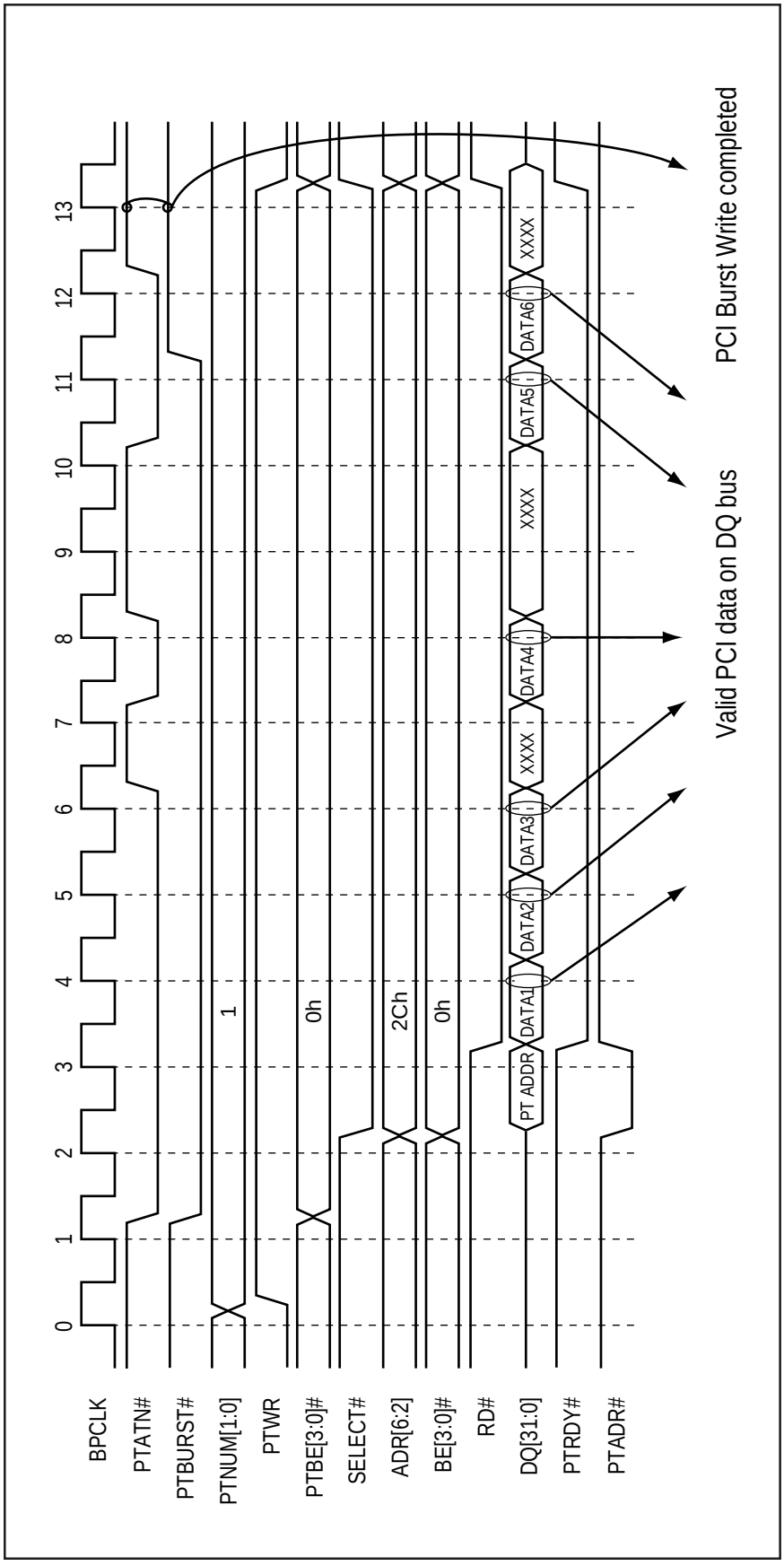


Figure 4. Pass-Thru Burst Write

PTBURST# Asserted. The access has a multiple data phases.

PTNUM[1:0] 01. Indicates the PCI access was to Pass-Thru region 1.

PTWR Asserted. The Pass-Thru access is a write.

PTBE[3:0]# 0h. Indicate the Pass-Thru access is 32-bits.

The PTADR# input is asserted to read the Pass-Thru Address Register. The byte enable, address, and SELECT# inputs are changed during this clock to select the Pass-Thru Data Register during clock cycle 3.

Clock 3: SELECT#, byte enables, and the address inputs remain driven to read the Pass-Thru Data Register at offset 2Ch. RD# is asserted to drive data register contents onto the DQ bus.

Clock 4: Add-On logic uses the rising edge of clock 4 to store DATA 1 from the S5933. PTRDY# asserted at the rising edge of clock 4 completes the current data phase. DATA 2 is driven on the Add-On bus.

Clock 5: Add-On logic uses the rising edge of clock 5 to store DATA 2 from the S5933. PTRDY# asserted at the rising edge of clock 5 completes the current data phase. DATA 3 is driven on the Add-On bus.

Clock 6: Add-On logic uses the rising edge of clock 6 to store DATA 3 from the S5933. PTRDY# asserted at the rising edge of clock 6 completes the current data phase. On the PCI bus, IRDY# has been deasserted, causing PTATN# to be deasserted. This is how a PCI initiator adds wait states, if it cannot provide data quickly enough. Data on the Add-On bus is not valid.

Clock 7: Because PTATN# remains deasserted, Add-On logic cannot store data at the rising edge of clock 7. PTATN# is reasserted, indicating the PCI initiator is no longer adding wait states. DATA 4 is driven on the Add-On bus.

Clock 8: Add-On logic uses the rising edge of clock 8 to store DATA 4 from the S5933. PTRDY# asserted at the rising edge of clock 8 completes the current data phase. On the PCI bus, IRDY# has been deasserted again, causing PTATN# to be deasserted. Data on the Add-On bus is not valid.

Clock 9: The PCI initiator is still adding wait states. Add-On logic cannot store data while PTATN# is deasserted.

Clock 10: Because PTATN# remains deasserted, Add-On logic cannot read data at the rising edge of clock 10. PTATN# is reasserted, indicating the PCI initiator is no longer adding wait states. DATA 5 is driven on the Add-On bus.

Clock 11: Add-On logic uses the rising edge of clock 11 to store DATA 5 from the S5933. PTRDY# asserted at the rising edge of clock 11 completes the current data phase. DATA 6 is driven on the Add-On bus.

Clock 12: Add-On logic uses the rising edge of clock 12 to store DATA 6 from the S5933. PTRDY# asserted at the rising edge of clock 12 completes the final data phase

Clock 13: PTATN# and PTBURST# deasserted at the rising edge of clock 13 indicates the Pass-Thru access is complete. The S5933 can accept new Pass-Thru accesses from the PCI bus at clock 15.

Figure 5 also shows a 5 data phase Pass-Thru burst write, but the Add-On logic uses PTRDY# to control the rate at which data is transferred. In many applications, Add-On logic is not fast enough to accept data at every BPCLK rising edge (every 30 ns in a 33 MHz PCI system). In this example, the Add-On interface accepts data every other clock. In the example, RD# is asserted during the entire Add-On burst, but it can be deasserted when PTRDY# is deasserted, the S5933 functions the same under both conditions.

Clock 0: PCI address information is stored in the S5933 Pass-Thru Address Register.

Clock 1: The PCI address is recognized as an access to Pass-Thru region 1. PCI data for the first data phase is stored in the S5933 Pass-Thru Data Register. PTATN# is asserted by the S5933 to indicate a Pass-Thru access is occurring.

Clock 2: Pass-Thru status signals indicate what action is required by Add-On logic. Pass-Thru status outputs are valid when PTATN# is active and are sampled by the Add-On at the rising edge of clock 2.

PTBURST# Asserted. The access has multiple data phases.

PTNUM[1:0] 01. Indicates the PCI access is to Pass-Thru region 1.

PTWR Asserted. The Pass-Thru access is a write.

PTBE[3:0]# 0h. Indicate the Pass-Thru access is 32-bits.

The PTADR# input is asserted to read the

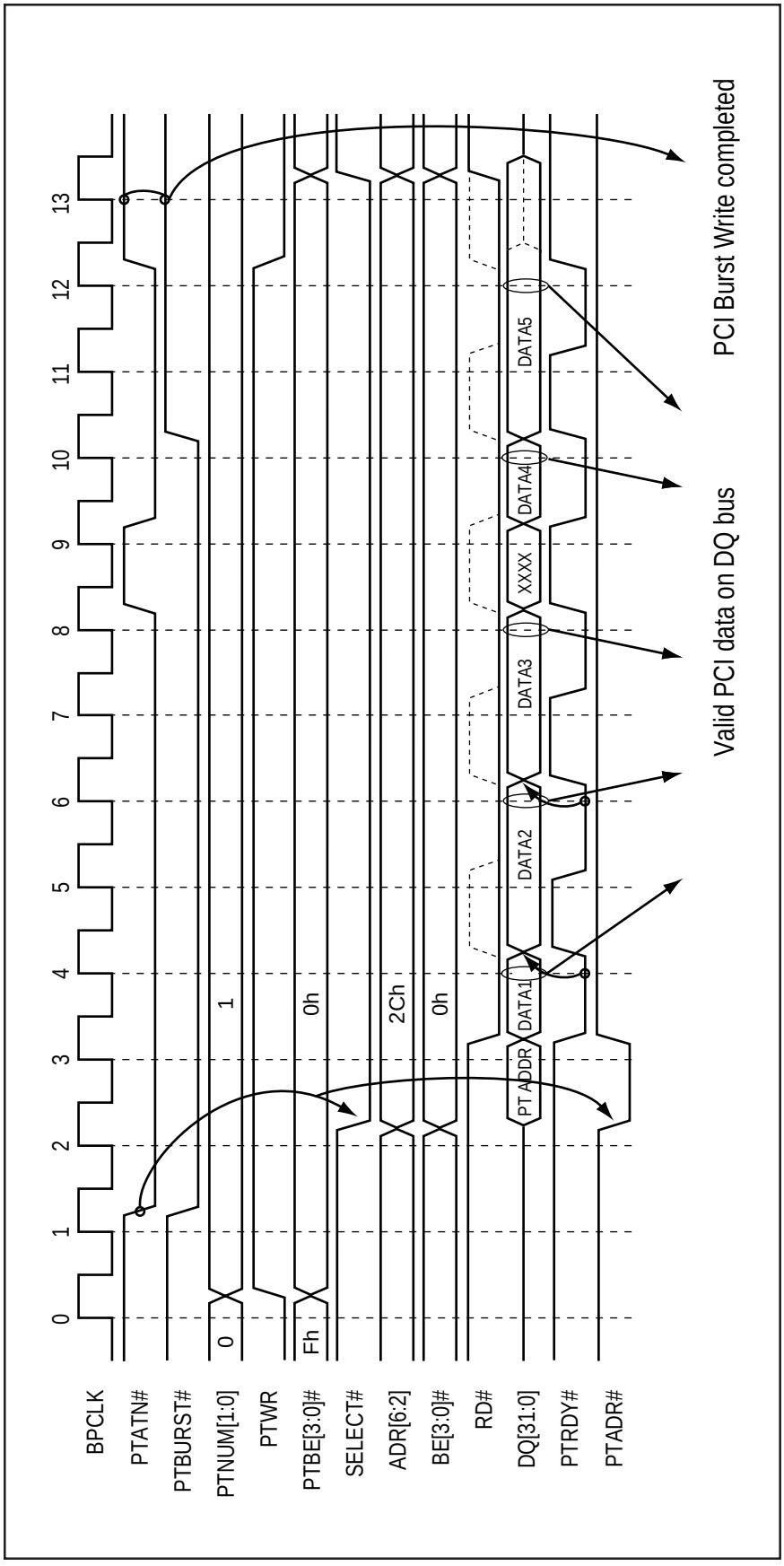


Figure 5. Pass-Thru Burst Writes Controlled by PTRDY#

Pass-Thru Address Register. The byte enable, address, and SELECT# inputs are changed during this clock to select the Pass-Thru Data Register during clock cycle 3.

Clock 3: SELECT#, byte enable, and the address inputs remain driven to read the Pass-Thru Data Register at offset 2Ch. RD# is asserted to drive data register contents onto the Add-On data bus.

Clock 4: Add-On logic uses the rising edge of clock 4 to store DATA 1 from the S5933. PTRDY# asserted at the rising edge of clock 4 completes the current data phase. DATA 2 is driven on the Add-On bus.

Clock 5: Add-On logic is not fast enough to store DATA 2 by the rising edge of clock 5. PTRDY# deasserted at the rising edge of clock 5 extends the current data phase and DATA 2 remains driven on the Add-On bus.

Clock 6: Add-On logic uses the rising edge of clock 6 to store DATA 2 from the S5933. PTRDY# asserted at the rising edge of clock 6 completes the current data phase. DATA 3 is driven on the Add-On bus.

Clock 7: Add-On logic is not fast enough to store DATA 3 by the rising edge of clock 7. PTRDY# deasserted at the rising edge of clock 7 extends the current data phase is and DATA 3 remains driven on the Add-On bus.

Clock 8: Add-On logic uses the rising edge of clock 8 to store DATA 3 from the S5933. PTRDY# asserted at the rising edge of clock 8 completes the current data phase. On the PCI bus, IRDY# has been deasserted, causing PTATN# to be deasserted. Data on the Add-On bus is not valid.

Clock 9: Because PTATN# remains deasserted, Add-On logic cannot store data at the rising edge of clock 9. PTATN# is reasserted, indicating the PCI initiator is no longer adding wait states. DATA 4 is driven on the Add-On bus

Clock 10: Add-On logic uses the rising edge of clock 10 to store DATA 4 from the S5933. PTRDY# asserted at the rising edge of clock 10 completes the current data phase. DATA 5 is driven on the Add-On bus. PTBURST# is deasserted, indicating that on the PCI bus, the burst is complete except for the last data phase. Since the data is double buffered, there may be one or two pieces of data available to the Add-On when PTBURST# becomes inactive. This example shows the single data available case. If another piece of data was available, then PTATN# would remain active instead of going inactive at clock 12.

Clock 11: Add-On logic is not fast enough to store DATA 5 by the rising edge of clock 11. PTRDY# deasserted at the rising edge of clock 11 extends the data phase and DATA 5 remains driven on the Add-On bus.

Clock 12: Add-On logic uses the rising edge of clock 12 to store DATA 5 from the S5933. PTRDY# asserted at the rising edge of clock 12 completes the final data phase.

Clock 13: PTATN# deasserted at the rising edge of clock 13 indicates the Pass-Thru access is complete. The S5933 can accept new Pass-Thru accesses from the PCI bus at clock 14.

Pass-Thru Burst Reads

A Pass-Thru burst read operation occurs when a PCI initiator reads multiple DWORDs from a Pass-Thru region. A burst transfer consists of a single address and a multiple data phases. During the address phase of the PCI transfer, the S5933 stores the PCI address into the Pass-Thru Address Register (APTA). If the S5933 determines that the address is within one of its defined Pass-Thru regions, it indicates to the Add-On that a write to the Pass-Thru Data Register (APTD) is required. Figure 6 shows a 6 data phase Pass-Thru burst read access (Add-On write) using PTADR#.

Clock 0: PCI address information is stored in the S5933 Pass-Thru Address Register. The PCI address is recognized as an access to Pass-Thru region 1. PTATN# is asserted by the S5933 to indicate a Pass-Thru access is occurring.

Clock 1: Pass-Thru status signals indicate what action is required by Add-On logic. Pass-Thru status outputs are valid when PTATN# is active and are sampled by the Add-On at the rising edge of clock 2.

PTBURST#	Deasserted, the S5933 does not yet recognize a PCI burst.
PTNUM[1:0]	01. Indicates the PCI access is to Pass-Thru region 1.
PTWR	Deasserted. The Pass-Thru access is a read.
PTBE[3:0]#	0h. Indicate the Pass-Thru access is 32-bits.

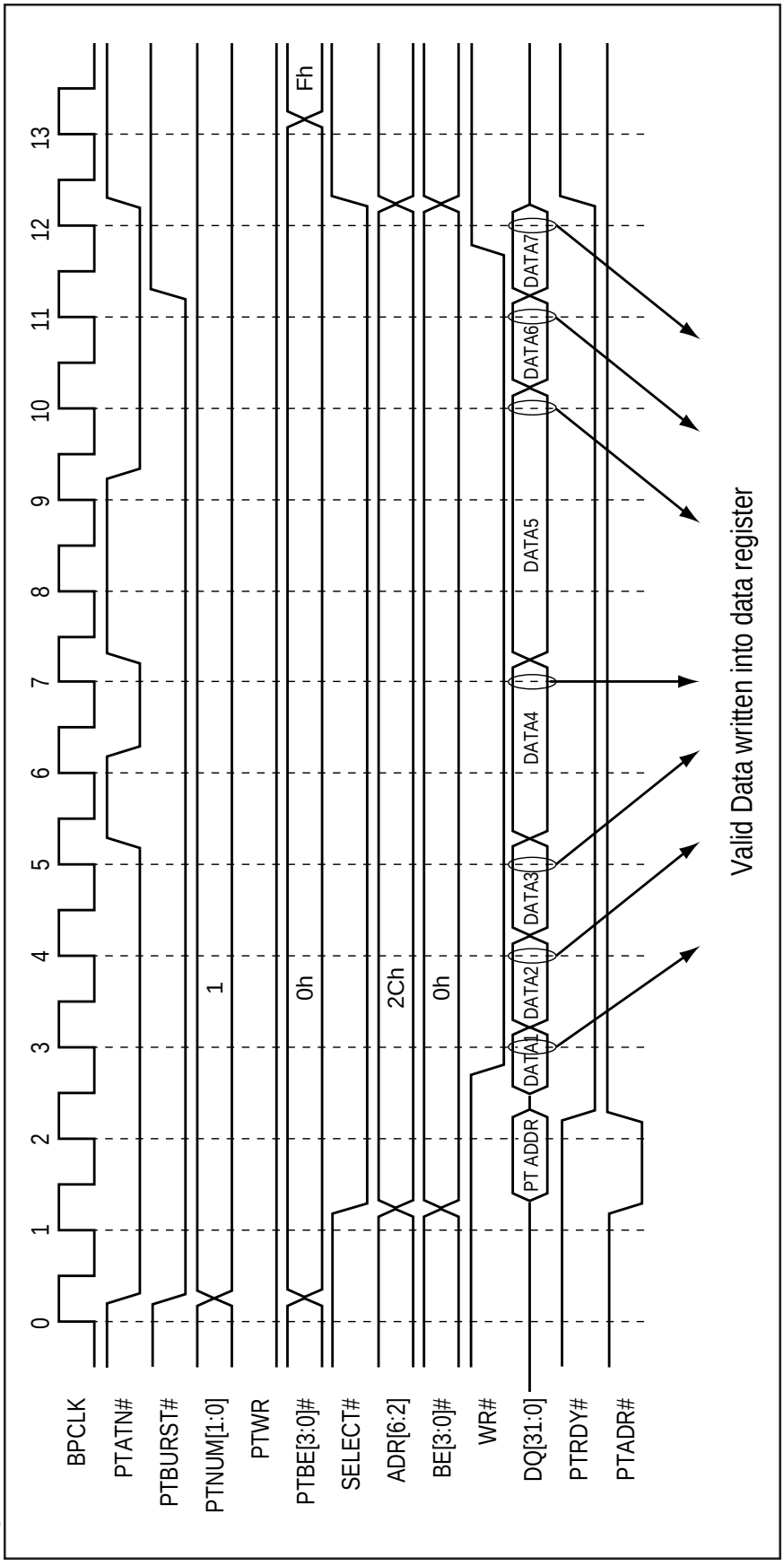


Figure 6. Pass-Thru Burst Read

The PTADR# input is asserted to read the Pass-Thru Address Register. The byte enable, address, and SELECT# inputs are changed during this clock to select the Pass-Thru Data Register during clock cycle 3.

Clock 2: SELECT#, byte enables, and the address inputs remain driven to read the Pass-Thru Data Register at offset 2Ch. PTBURST# is asserted by the S5933, indicating the current Pass-Thru read is a burst.

Clock 3: WR# asserted at the rising edge of clock 3 writes DATA 1 into the S5933. PTRDY# asserted at the rising edge of clock 3 completes the current data phase.

Clock 4: WR# asserted at the rising edge of clock 4 writes DATA 2 into the S5933. PTRDY# asserted at the rising edge of clock 4 completes the current data phase.

Clock 5: WR# asserted at the rising edge of clock 5 writes DATA 3 into the S5933. PTRDY# asserted at the rising edge of clock 5 completes the current data phase. On the PCI bus, IRDY# has been deasserted, causing PTATN# to be deasserted. This is how a PCI initiator adds wait states, if it cannot read data quickly enough.

Clock 6: PTATN# remains deasserted at the rising edge of clock 6. The Add-On cannot write DATA 4 until PTATN# is asserted. PTATN# is reasserted during the cycle, indicating the PCI initiator is no longer adding wait states. Add-On logic continues to drive DATA 4 on the Add-On bus.

Clock 7: WR# asserted at the rising edge of clock 7 writes DATA 4 into the S5933. PTRDY# asserted at the rising edge of clock 7 completes the current data phase. On the PCI bus, IRDY# has been deasserted, causing PTATN# to be deasserted. The PCI initiator is adding wait states.

Clock 8: PTATN# remains deasserted at the rising edge of clock 8. The Add-On cannot write DATA 5 until PTATN# is asserted. Add-On logic continues to drive DATA 5 on the Add-On bus.

Clock 9: PTATN# remains deasserted at the rising edge of clock 9. The Add-On cannot write DATA 5 until PTATN# is asserted. Add-On logic continues to drive DATA 5 on the Add-On bus. PTATN# is reasserted during the cycle, indicating the PCI initiator is done adding wait states.

Clock 10: WR# asserted at the rising edge of clock 10 writes DATA 5 into the S5933. PTRDY# asserted at the rising edge of clock 10 completes the current data phase.

Clock 11: WR# asserted at the rising edge of clock 11 writes DATA 6 into the S5933. PTRDY# asserted at the rising edge of clock 11 completes the final data phase.

Clock 12: PTBURST# is deasserted at the rising edge of clock 12 indicating the Pass-Thru burst is complete. The S5933 can accept new Pass-Thru accesses from the PCI bus at clock 14. Any data written into the Pass-Thru data register is not required and is never passed to the PCI interface (as PTRDY# is not asserted at the rising edge of clock 13).

Figure 7 also shows a 5 data phase Pass-Thru burst read, but the Add-On logic uses PTRDY# to control the rate at which data is transferred. In many applications, Add-On logic is not fast enough to provide data every BPCLK (every 30 ns in a 33 MHz PCI system). In this example, the Add-On interface writes data every other clock cycle. WR# is shown asserted during the entire Add-On burst, but WR# can be deasserted when PTRDY# is deasserted, the S5933 functions the same under both conditions.

Clock 0: PCI address information is stored in the S5933 Pass-Thru Address Register. The PCI address is recognized as an access to Pass-Thru region 1. PTATN# is asserted by the S5933 to indicate a Pass-Thru access is occurring.

Clock 1: Pass-Thru status signals indicate what action is required by Add-On logic. Pass-Thru status outputs are valid when PTATN# is active and are sampled by the Add-On at the rising edge of clock 2.

PTBURST#	Deasserted, the S5933 does not yet recognize a PCI burst.
PTNUM[1:0]	01. Indicates the PCI access is to Pass-Thru region 1.
PTWR	Deasserted. The Pass-Thru access is a read.
PTBE[3:0]#	0h. Indicate the Pass-Thru access is 32-bits.

The PTADR# input is asserted to read the Pass-Thru Address Register. The byte enable, address, and SELECT# inputs are changed during this clock to select the Pass-Thru Data Register during clock cycle 3.

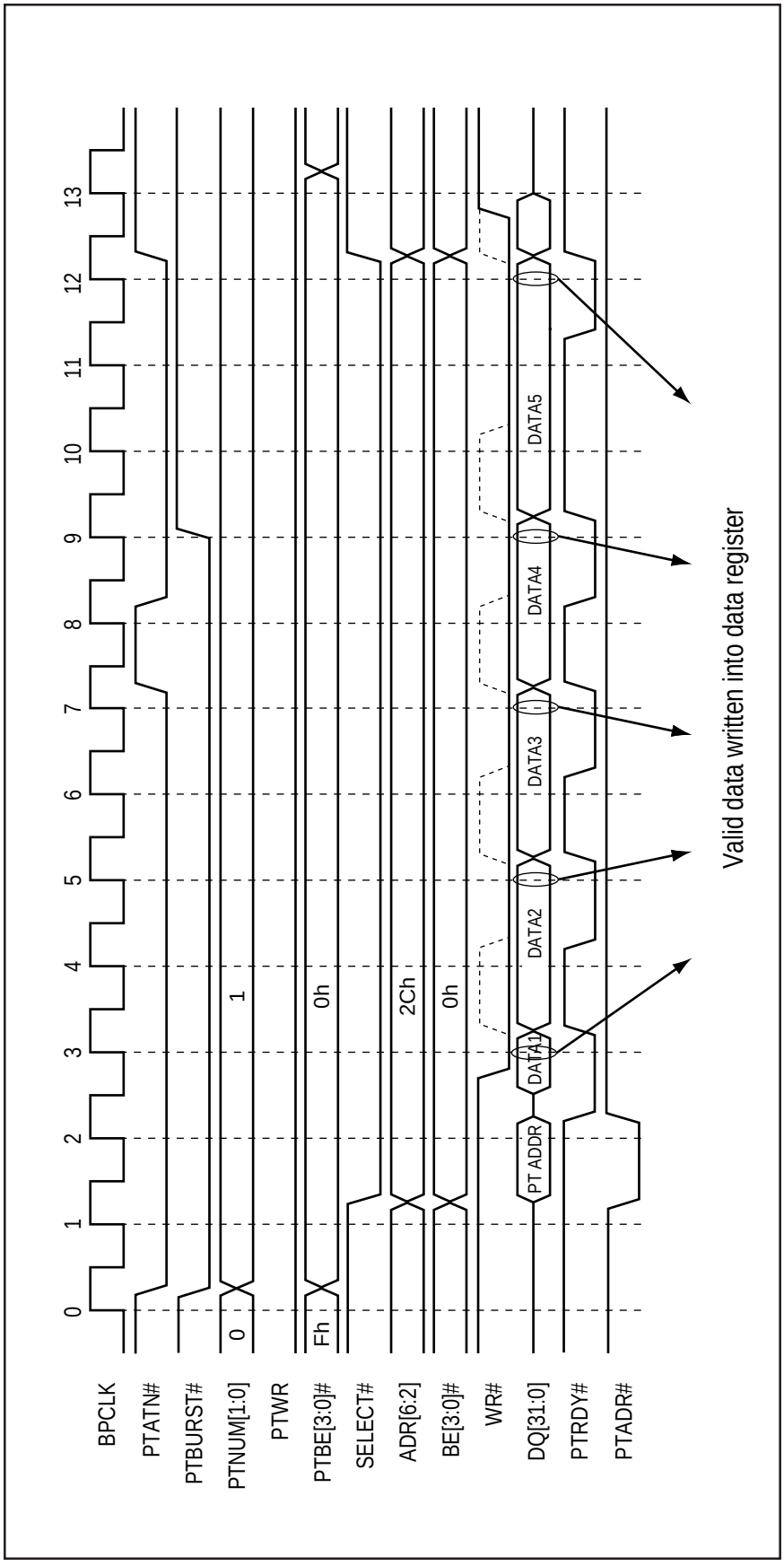


Figure 7. PCI Burst Read Controlled by PTRDY#

Clock 2: SELECT#, byte enable, and the address inputs remain driven to read the Pass-Thru Data Register at offset 2Ch. PTBURST# is asserted by the S5933, indicating the current Pass-Thru read is a burst.

Clock 3: WR# asserted at the rising edge of clock 3 writes DATA 1 into the S5933. PTRDY# asserted at the rising edge of clock 3 completes the current data phase.

Clock 4: Add-On logic drives DATA 2 on the Add-On bus, but PTRDY# deasserted at the rising edge of clock 4 extends the current data phase.

Clock 5: WR# asserted at the rising edge of clock 5 writes DATA 2 into the S5933. PTRDY# asserted at the rising edge of clock 5 completes the current data phase.

Clock 6: Add-On logic drives DATA 3 on the Add-On bus, but PTRDY# deasserted at the rising edge of clock 6 extends the current data phase.

Clock 7: WR# asserted at the rising edge of clock 7 writes DATA 3 into the S5933. PTRDY# asserted at the rising edge of clock 7 completes the current data phase. On the PCI bus, IRDY# has been deasserted, causing PTATN# to be deasserted. This is how a PCI initiator adds wait states, if it cannot read data quickly enough.

Clock 8: PTATN# remains deasserted at the rising edge of clock 8. The Add-On cannot write DATA 4 until PTATN# is asserted. Add-On logic continues to drive DATA 4 on the Add-On bus. PTATN# is reasserted during the cycle, indicating the PCI initiator is done adding wait states.

Clock 9: WR# asserted at the rising edge of clock 9 writes DATA 4 into the S5933. PTRDY# asserted at the rising edge of clock 9 completes the current data phase.

Clock 10: Add-On logic drives DATA 5 on the Add-On bus, but PTRDY# deasserted at the rising edge of clock 10 extends the current data phase.

Clock 11: PTATN# remains deasserted at the rising edge of clock 11. The Add-On does not have to write DATA 5 until PTATN# is asserted. Add-On logic continues to drive DATA 5 on the Add-On bus. PTATN# is reasserted during the cycle, indicating the PCI initiator is done adding wait states.

Clock 12: PTRDY# asserted at the rising edge of clock 12 completes the final data phase. Any data written into the Pass-Thru data register is not required and is never passed to the PCI interface (as PTRDY# is not asserted at the rising edge of clock 13).

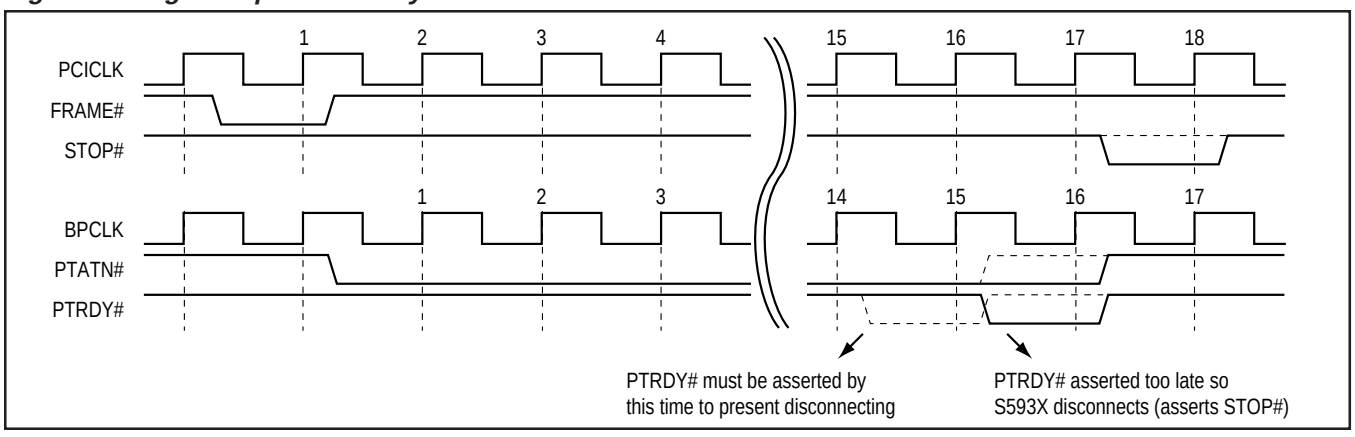
Clock 13: PTATN# and PTBURST# deasserted at the rising edge of clock 13 indicates the Pass-Thru access is complete. The S5933 can accept new Pass-Thru accesses from the PCI bus at clock 14.

Add-On Pass-Thru Disconnect Operation

Slow PCI targets are prevented from degrading PCI bus performance. The PCI specification allows only 16 clocks for a target to respond before it must request a retry on single data phase accesses. For burst accesses, the first data phase is allowed 16 clocks to complete, all subsequent data phases are allowed 8 clocks each. This requirement allows other PCI initiators to use the bus while the target requesting the retry completes the original access.

Figure 8 shows the conditions that cause the S5933 to request a retry from a PCI initiator on the first data phase of a PCI read operation. FRAME# is asserted during the rising edge of PCI clock 1. From this point,

Figure 8. Target Requested Retry on the First PCI Data Phase



the S5933 has 16 clock cycles to respond to the initiator with TRDY# (completing the cycle). FRAME# could remain asserted, indicating a burst read, but the retry request conditions are identical for a single data phase read and the first data phase of a burst read.

BPCLK is identical to PCICLK, lagging by a propagation delay of a few nanoseconds (see Chapter 13). PTATN# is asserted on the Add-On interface as soon as FRAME# is sampled active at a PCICLK rising edge.

After PTATN# is asserted, PTRDY# must be asserted by the 15th BPCLK rising edge to prevent the S5933 from requesting a retry. TRDY# is asserted on the PCI interface one clock cycle after PTRDY# is asserted on the Add-On interface. If Add-On logic does not return PTRDY# by the 15th BPCLK rising edge, the S5933 asserts STOP#, requesting a retry from the PCI initiator.

For Pass-Thru write operations, the S5933 never disconnects on the first or second PCI data phases of a burst. The first data and second phases are always accepted immediately by the S5933. No further action is required by the PCI initiator. The only situation where the S5933 may respond to a Pass-Thru write with a retry request is after the second data phase of a Pass-Thru burst write.

Figure 9 shows the conditions required for the S5933 to request a retry after the second data phase of a burst transfer. This figure applies to both Pass-Thru burst read and write operations.

The previous data phase is completed with the assertion of PTRDY# at the rising edge of BPCLK 0. Add-On logic must assert PTRDY# by the rising edge of BPCLK 8 to prevent the S5933 from asserting STOP#, requesting a retry. Meeting this condition allows the S5933 to assert TRDY# by the rising edge of PCICLK 8, completing the data phase with requiring a retry.

When the S5933 requests a retry, the Pass-Thru status indicators remain valid (allowing the Add-On logic to complete the access). PTBURST# is the exception to this. PTBURST# is deasserted to indicate that there is currently no burst in progress on the PCI bus. The other Pass-Thru status indicators remain valid until PTATN# is deasserted. Figure 10 shows the Add-On bus interface signals after the S5933 requests a retry.

As long as PTATN# remains asserted, Add-On logic should continue to transfer data. For PCI read operations, one Add-On write operation is required after a retry request. After the Add-On write, asserting PTRDY# deasserts PTATN#.

For Pass-Thru write operations, one or two data transfers may remain after the S5933 signals a retry. Two data transfers are possible because the S5933 has a double buffered Pass-Thru data register used for writes. A PCI burst may have filled both registers before the S5933 requested a retry. PTATN# remains asserted until both are emptied. PTRDY# must be asserted after each read from the Pass-Thru data register. If both registers are full, PTATN# is deasserted only after PTRDY# is asserted the second time. The S5933 only accepts further PCI accesses after both registers are emptied.

8-Bit and 16-Bit Pass-Thru Add-On Bus Interface

The S5933 allows a simple interface to devices with 8-bit or 16-bit data buses. Each Pass-Thru region may be defined as 8-, 16-, or 32-bits, depending on the contents of the nv memory boot device loaded into the PCI Base Address Configuration Registers during initialization. The Pass-Thru Add-On interface internally controls byte lane steering to allow access to the 32-bit Pass-Thru Data Register (APTD) from 8-bit or 16-bit Add-On buses.

Figure 9. Target Requested Retry after the First Data Phase of a Burst Operation

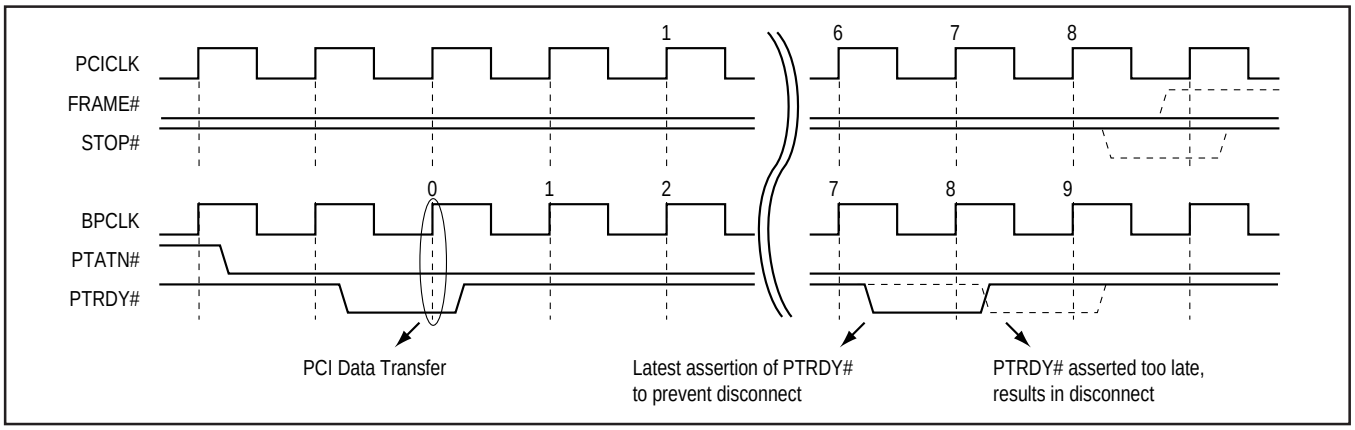
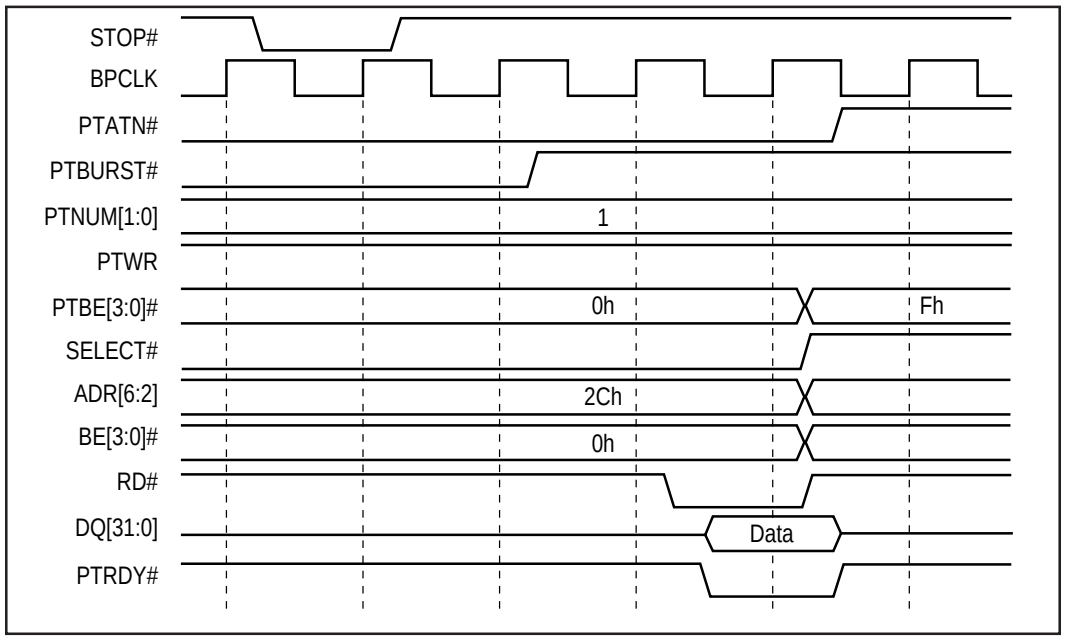


Figure 10. Pass-Thru Signals after a Target Requested Retry



Internal byte lane steering may be used whether the MODE input defines a 16-bit or 32-bit Add-On interface. When a 16-bit Add-On interface is used, the ADR1 input is used in conjunction with the byte enables to steer data into the proper APTD register byte locations.

If MODE defines a 16-bit interface, only 16-bits of address are driven when PTADR# is asserted. If more than 16-bits of address are required, the Pass-Thru Address Register (APTA) must be read with SELECT#, RD#, byte enable and address inputs. Two consecutive reads are required to latch all of the address information (one with ADR1=0, one with ADR1=1).

Regardless of MODE, various data widths may be used. For Pass-Thru writes (Add-On APTD reads), Add-On logic must read the APTD register one byte or one word at a time (depending on the Add-On bus width). The internal data bus is steered to the correct portion of APTD using the BE[3:0]# inputs. Table 1 shows the byte lane steering mechanism used by the S5933. The BYTEn symbols indicate data bytes in the Pass-Thru Data Register.

When a read is performed with a BEn# input asserted, the corresponding PTBEn# output is deasserted. Add-On logic cycles through the byte enables to read the entire APTD register. Once all data is read (PTBE[3:0]# are deasserted), PTRDY# is asserted by the Add-On, completing the access.

For Pass-Thru reads (Add-On APTD writes), the bytes requested by the PCI initiator are indicated by the PTBE[3:0]# outputs. Add-On logic uses the PTBE[3:0]# signals to determine which bytes must be written (and which bytes have already been written). For example, a

Table 1. Byte Lane Steering for Pass-Thru Data Register Read (PCI Write)

Byte Enables				APTD Register Read Byte Lane Steering			
3	2	1	0	DQ[31:24]	DQ[23:16]	DQ[15:8]	DQ[7:0]
x	x	x	0	BYTE3	BYTE2	BYTE1	BYTE0
x	x	0	1	BYTE3	BYTE2	BYTE1	BYTE1
x	0	1	1	BYTE3	BYTE2	BYTE3	BYTE2
0	1	1	1	BYTE3	BYTE3	BYTE3	BYTE3

PCI initiator performs a byte Pass-Thru read from an 8-bit Pass-Thru region with PCI BE2# asserted. On the Add-On interface, PTBE2# is asserted, indicating that the PCI initiator requires data in this byte. Once the Add-On writes APTD, byte 2, PTBE2# is deasserted, and the Add-On may assert PTRDY#, completing the cycle.

Table 2 shows how the external Add-On data bus is steered to the Pass-Thru Data Register bytes. This mechanism is determined by the Pass-Thru region bus width defined during initialization (see Section 12.3). The BYTEn symbols indicate data bytes in the Pass-Thru Data Register. For example, an 8-bit Add-On write with BE1# asserted results in the data on DQ[7:0] being steered into BYTE1 of the APTD register.

Table 2. Byte Lane Steering for Pass-Thru Data Register Write (PCI Read)

Defined	APTD Register Write Byte Lane Steering			
PT-Bus Width	BYTE3	BYTE2	BYTE1	BYTE0
32-Bit Data Bus	DQ[31:24]	DQ[23:16]	DQ[15:8]	DQ[7:0]
16-Bit Data Bus	DQ[15:8]	DQ[7:0]	DQ[15:8]	DQ[7:0]
8-Bit Data Bus	DQ[7:0]	DQ[7:0]	DQ[7:0]	DQ[7:0]

To write data into the APTD Register, the PTBEn# output and the BEn# input must both be asserted. The following describes how APTD Register writes are controlled:

Write BYTE3 if PTBE3# AND BE3# are asserted

Write BYTE2 if PTBE2# AND BE2# are asserted

Write BYTE1 if PTBE1# AND BE1# are asserted

Write BYTE0 if PTBE0# AND BE0# are asserted

After each byte is written into the Pass-Thru data register, its corresponding PTBE[3:0]# output is deasserted. This allows Add-On logic to monitor which bytes have been written, and which bytes remain to be written. When all bytes requested by the PCI initiator have been written, the PTBE[3:0]# are all be deasserted, and the Add-On asserts PTRDY#.

Figure 11 shows Pass-Thru operation for a region defined for an 8-bit Add-On bus interface. As the 8-bit device is connected only to DQ[7:0], the device must access APTD one byte at a time.

The PCI initiator has performed a 32-bit write of 08D49A30h to Pass-Thru region zero. PTBE[3:0]# are all asserted. At clock 1, the Add-On begins reading the APTD Register (asserting SELECT#, ADR[6:2], and RD#). Add-On logic asserts BE0#, and BYTE0 of APTD is driven on DQ[7:0]. At the rising edge of clock 2, BE0# is sampled by the S5933 and PTBE0# is deasserted. PTBE[3:1]# are still asserted.

During clock 2, only BE1# is activated, and BYTE1 of APTD is driven on DQ[7:0]. At the rising edge of clock 3, BE1# is sampled by the S5933 and PTBE1# is deasserted. PTBE[3:2]# are still asserted.

This process continues until all bytes have been read from the APTD Register. During clock 5, RD# is deasserted and PTRDY# is asserted. PTRDY# is sampled by the S5933 at the rising edge of clock 6, and the current data phase is completed. PTATN# is deasserted and new data can be written from the PCI bus. In this example, the byte enables are asserted, sequentially, from BE0# to BE3#. This is not required, bytes may be accessed in any order.

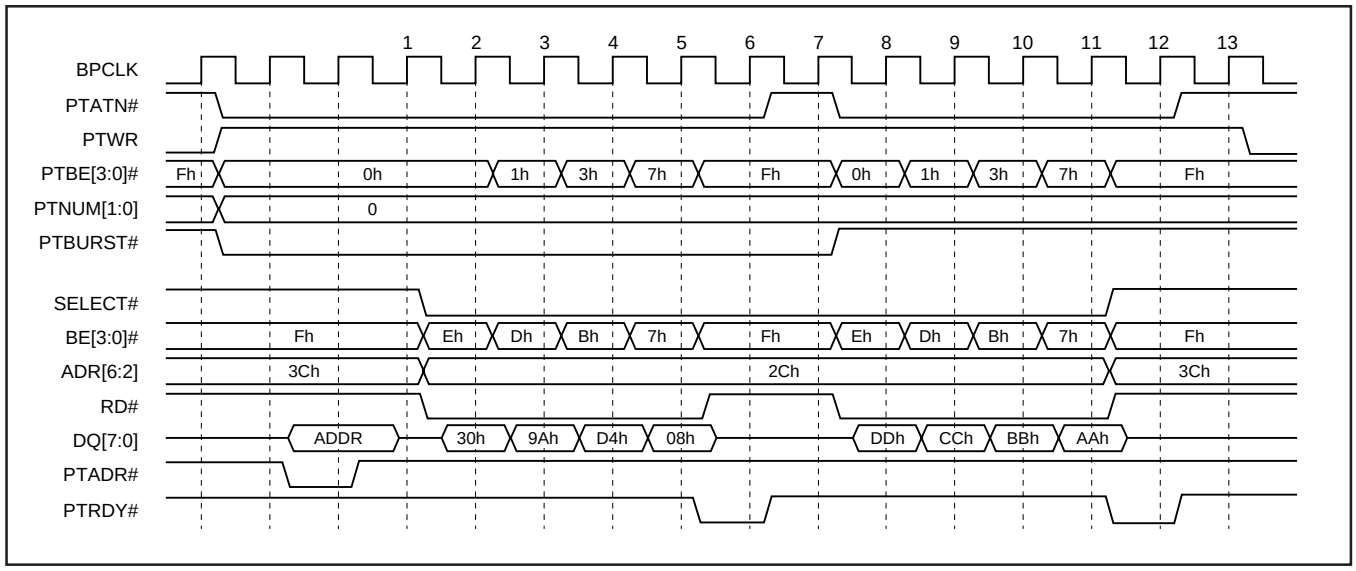
New data is written by the PCI initiator and is available in the APTD Register during clock 7. RD# is asserted and the byte enables are cycled again. With each new data from the PCI bus, the Add-On sequences through the byte enables to access APTD via DQ[7:0].

For 16-bit peripheral devices, the byte steering works in the same way. Because the Add-On data bus is 16-bits wide, only two 16-bit cycles are required to access the entire APTD Register. Two byte enables can be asserted during each access.

In Figure 11, RD# is held low and the byte enables are changed each clock. This assumes the Add-On can accept data at one byte per clock. This is the fastest transfer possible. For slower devices, wait states may be added.

As long as the byte enables remain in a given state, the corresponding byte of the APTD Register is connected to the DQ bus (the RD# or WR# pulse may also be lengthened). Each access may be extended for slower Add-On devices, but extending individual data phases for Pass-Thru cycles may result in the S5933 requesting retries by the initiator.

Figure 11. Pass-Thru Write to an 8-bit Add-On Device



Note: 8 Bit Mode BE's are E, D, B, 7; 16 Bit Mode BE's are C, 3.

CONFIGURATION

The S5933 Pass-Thru interface utilizes four Base Address Registers (BADR1:4). Each Base Address Register corresponds to a Pass-Thru region. The contents of these registers during initialization determine the characteristics of that particular Pass-Thru region. Each region can be mapped to memory or I/O space. Memory mapped devices can, optionally, be mapped below 1 Mbyte and can be identified as prefetchable. Both memory and I/O regions can be configured as 8-, 16-, or 32-bits wide.

The designer has the option to use 1, 2, 3, 4 or none of the Pass-Thru regions. Base Address Registers are loaded during initialization from the external non-volatile boot device. Without an external boot device, the default value for the BADR registers is zero (region disabled). The Base Address Registers are the only registers that define Pass-Thru operation.

S5933 Base Address Register Definition

Some bits in the Base Address Registers have specific functions. The following bits have special functions:

D0 Memory or I/O mapping. If this bit is clear, the region should be memory mapped. If this bit is set, the region should be I/O mapped.

D2:1 Location of a memory region. These bits request that the region be mapped in a particular part of memory. These bit definitions are only used for memory mapped regions.

D2	D1	Location
0	0	Anywhere in 32-bit memory space
0	1	Below 1 Mbyte in memory space (Real Mode address space)
1	0	Anywhere in 64-bit memory space (not valid for the S5933)
1	1	Reserved

D3 Prefetchable. For memory mapped regions, the region can be defined as cacheable. If set, the region is cacheable. If this bit is clear, the region is not.

D31:30 Pass-Thru region bus width. These two bits are used by the S5933 to define the data bus width for a Pass-Thru region. Regardless of the programming of other bits in the BADR register, if D31:30 are zeros, the Pass-Thru region is disabled.

D31	D30	Add-On Bus Width
0	0	Region disabled
0	1	8-bits
1	0	16-bits
1	1	32-bits

BADR1:4 bits D31:30 are used only by the S5933. When the host reads the Base Address Registers during configuration cycles, they always return the same value as D29. If D29 is zero, D31:30 return zero, indicating the region is disabled. If D29 is one, D31:30 return one. This operation limits each Pass-Thru region to a maximum size of 512 Mbytes of memory.

For I/O mapped regions, the PCI specification allows no more than 256 bytes per region. The S5933 allows larger regions to be requested by the Add-On, but a PCI BIOS will not allocate the I/O space and will probably disable the region.

Creating a Pass-Thru Region

Page 3-40 describes the values that must be programmed into the non-volatile boot device to request various block sizes and characteristics for Pass-Thru regions. After reset, the S5933 downloads the contents of the boot device locations 54h, 58h, 5Ch, and 60h into “masks” for the corresponding Base Address Registers. The following are some examples for various Pass-Thru region definitions:

NV Memory Contents	Pass-Thru Region Definition
54h = BFFFF002h	Pass-Thru region 1 is a 4Kbyte region, mapped below 1 Mbyte in memory space with a 16-bit Add-On data bus. This memory region is not cacheable.
58h = 3xxxxxxxh	Pass-Thru region 2 is disabled. (D31:30 = 00.)
60h = FFFFFF81h	Pass-Thru region 3 is a 32-bit, 128 byte I/O-mapped region.
64h = 00000000h	Pass-Thru region 4 is disabled.

During the PCI bus configuration, the host CPU writes all ones to each Base Address Register, and then reads the contents of the registers back. The mask downloaded from the boot device determines which bits are read back as zeros and which are read back as ones. The number of zeros read back indicates the amount of memory or I/O space a particular S5933 Pass-Thru region is requesting.

After the host reads all Base Address Registers in the system (as every PCI device implements from one to six), the PCI BIOS allocates memory and I/O space to each Base Address region. The host then writes the start address of each region back into the Base Address Registers. The start address of a region is always an integer multiple of the region size. For example, a 64 Kbyte memory region is always mapped to begin on a 64K boundary in memory. It is important to note that no PCI device can xbe absolutely located in system memory or I/O space. All mapping is determined by the system, not the application.

Accessing a Pass-Thru Region

After the system is finished defining all Base Address Regions within a system, each Base Address Register contains a physical address. The application software must now find the location in memory or I/O space of its hardware. PCI systems provide BIOS or operating system function calls for application software to find particular devices on the PCI bus based on Vendor ID and Device ID values. This allows application software to access the device's Configuration Registers.

The Base Address Register values in the S5933's Configuration Space may then be read and stored for use by the program to access application hardware. The value in the Base Address Registers is the physical address of the first location of that Pass-Thru region. Some processor architectures allow this address to be used directly to access the PCI device. For Intel Architecture systems, the physical address must be changed into a Segment/Offset combination.

For Real Mode operation in an Intel Architecture system (device mapped below 1 Mbyte in memory), creating a Segment/Offset pair is relatively simple. To calculate a physical address, the CPU shifts the segment register 4 bits to the left and adds the offset (resulting in a 20 bit physical address). The value in the Base Address Register must be read and shifted 4 bits to the right. This is the segment value and should be stored in one of the Segment registers. An offset of zero (stored in SI, DI or another offset register) accesses the first location in the Pass-Thru region.

ABSOLUTE MAXIMUM RATINGS

.

Parameter	Min	Max	Units
Storage Temperature	–55	125	°C
Supply Voltage (V_{CC})	–0.3	7.0	Volts
Input Pin Voltage	–0.5	V_{CC}+ 5.0	Volts
Power Dissipation		1.05	Watts @ 33 MHz

DC CHARACTERISTICS

The Following table summarizes the required parameters defined by the PCI specification as they apply to the S5933 controller.

PCI Input/Output Electrical Characteristics

Symbol	Parameter	Min	Max	Units	Test Conditions	Notes
V_{CC}	Supply Voltage	4.75	5.25	V		
V_{ih}	Input High Voltage	2.0		V		
V_{il}	Input Low Voltage	–0.5	0.8	V		
I_{ih}	Input High Leakage Current		70	µA	V_{in} = 2.7	1
I_{il}	Input Low Leakage Current		–70	µA	V_{in} = 0.5	1
V_{oh}	Output High Voltage	2.4		V	I_{out} = –2mA	
V_{ol}	Output Low Voltage		0.55	V	I_{out} = 3mA, 6mA	2
C_{in}	Input Pin Capacitance		10	pF		3
C_{clk}	CLK Pin Capacitance	5	12	pF		
C_{IDSEL}	IDSEL Pin Capaticance		8	pF		

Notes:

1. Input leakage applies to all inputs and bi-directional buffers.
2. PCI Bus signals without pull-up resistors will provide the 3 mA output current. Signals which require a pull-up resistor will provide 6 mA output current.
3. The PCI specification limits all PCI inputs not located on the motherboard to 10 pf (the clock is allowed to be 12 pf).

PCI Bus Signals

The following table summarizes the PCI Bus DC parameters defined by the PCI specification as they apply to the S5933 controller.

Signal	Type	Direction	Max	Units	Notes
CLK		Input			
RST#		Input			
INTA#	Open Drain	Output	4	mA	
AD[31:0]	t/s	Bi-directional		mA	
REQ#	t/s	Output	4	mA	
GNT#		Input			
C/BE[3:0]#	t/s	Bi-directional	4	mA	
DEVSEL#	s/t/s	Bi-directional		mA	
FRAME#	s/t/s	Bi-directional	4	mA	
IRDY#	s/t/s	Bi-directional	4	mA	
TRDY#	s/t/s	Bi-directional	4	mA	
PERR#	s/t/s	Bi-directional	4	mA	
PAR	t/s	Bi-directional	4	mA	
SERR#	Open Drain	Output	4	mA	
STOP#	s/t/s	Bi-directional	4	mA	
LOCK#		Input			
IDSEL		Input			

Add-On Bus Signals

The following table summarizes the Add-On Bus DC parameters as they apply to the S5933 controller.

Signal	Type	Direction	Max	Units	Notes
PCLK		Output	8	mA	
IRQ#		Output	4	mA	
SYSRST#		Output	4	mA	
ADR[6:2]		Input			
SELECT		Input			
ADR[6:2]		Input			
BE[3:0]#		Input			
RD#		Input			
WR#		Input			
DQ[31:0]	t/s	Bi-directional	4	mA	
WRFULL		Output	4	mA	
RDEMPTY		Output	4	mA	
RDFIFO#		Input			
WRFIFO#		Input			
PTATN#		Output	4	mA	
PTBURST#		Output	4	mA	
PTADR#		Input			
PTRDY#		Input			
PTWR		Output	4	mA	
PTBE[3:0]#		Output	4	mA	
PTNUM[1:0]		Output	4	mA	
EQ[7:0]	t/s	Bi-directional	1	mA	
EA[8:0]	t/s	Output	1	mA	
EA[15:9]		Output	1	mA	
MODE		Input			
TEST		Output	4	mA	
FLT#		Input			
ERD#/SCL		Output	1	mA	
EWR#/SDA	t/s	Bi-directional	1	mA	

AC CHARACTERISTICS

PCI Bus Timings

Functional Operation Range (V_{CC}=5.0V ±5%, 0˚C to 70˚C, 50 pF load on outputs)

Symbol	Parameter	Min	Max	Units	Notes
TCL	Cycle Time	30		ns	
t1	High Time	12		ns	
t2	Low Time	12		ns	
t3	Rise Time (0.8V to 2.0V)		3	ns	
t4	Fall Time (2.0V to 0.8V)		3	ns	
t5	Output Valid Delay (Bussed Signals) Output Valid Delay (Point-to-Point Signals)	2 2	11 12	ns	Note 1
t6	Float to Active Delay	2		ns	
t7	Active to Float Delay		28	ns	
t8	Rising Edge Setup (Bussed Signals) Rising Edge Setup (GNT#) Rising Edge Setup (REQ#)	7 10 12		ns	
t9	Hold from PCI Clock Rising Edge	0		ns	

Notes:

1. Minimum times are for unloaded outputs, maximum times are for 50 pF equivalent loads.

Figure 1. PCI Clock Timing

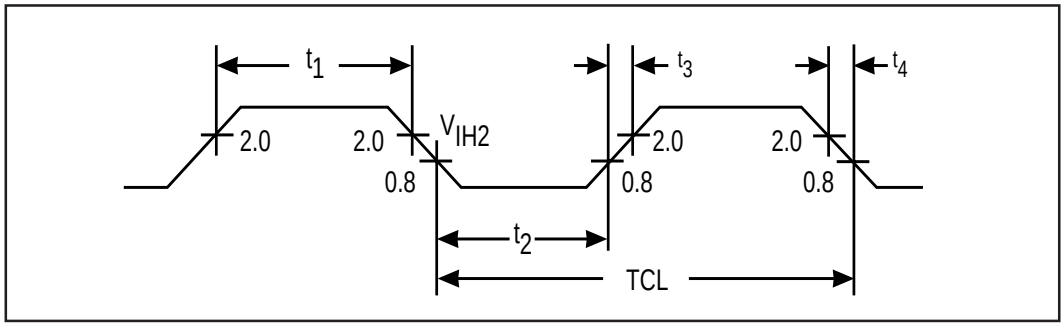


Figure 2. PCI Output Timing

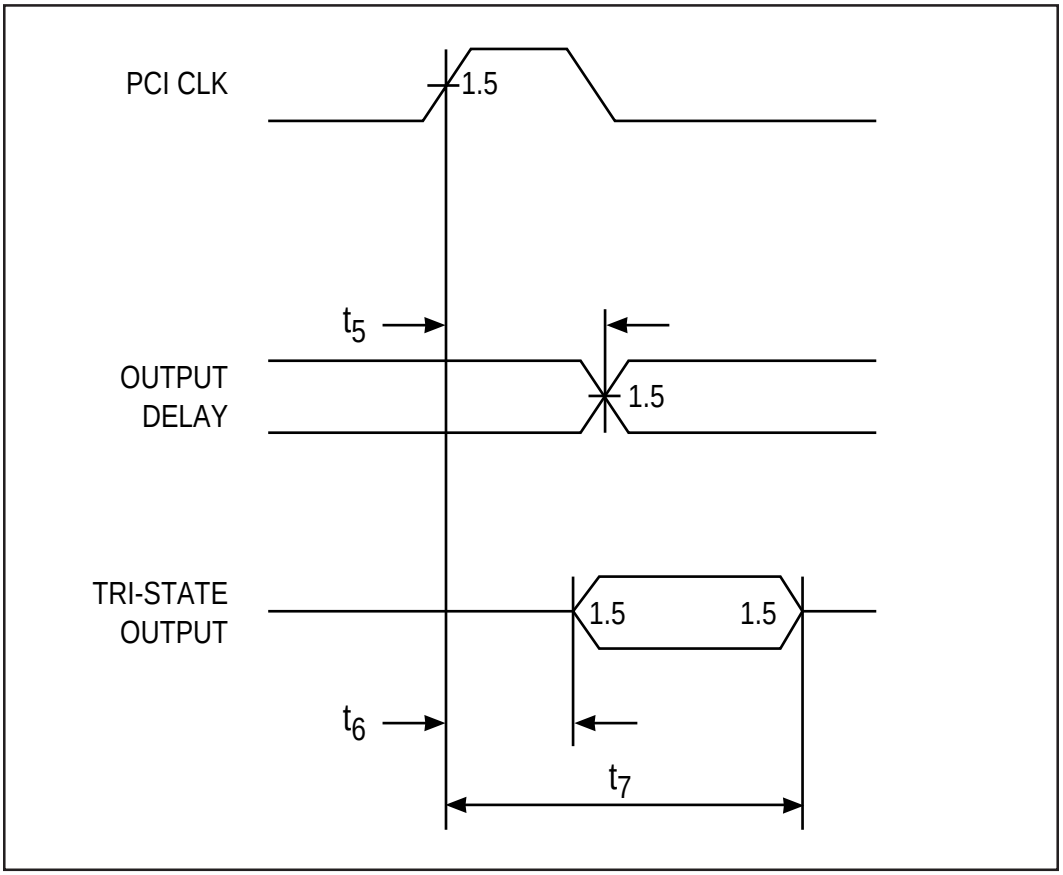
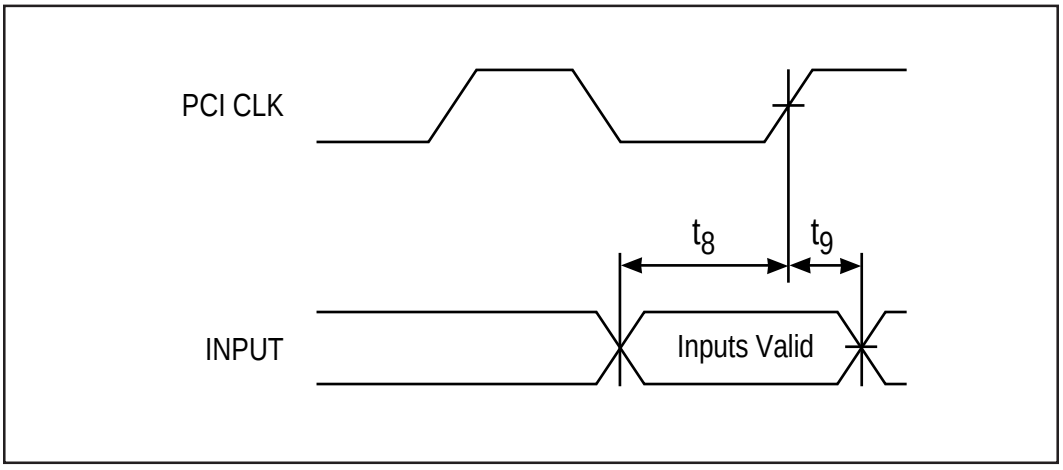


Figure 3. PCI Input Timing



Add-On Bus Timings

Figure 4. Add-On Clock Timing

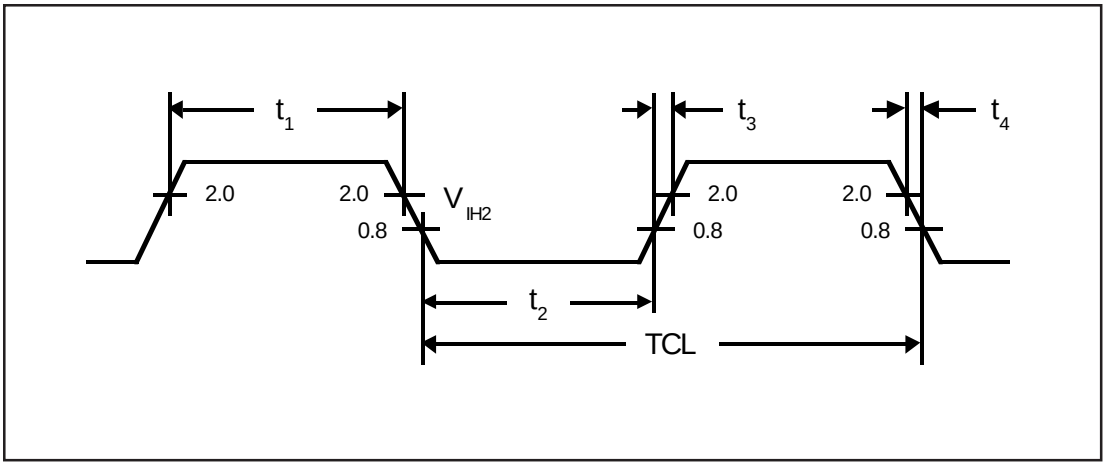
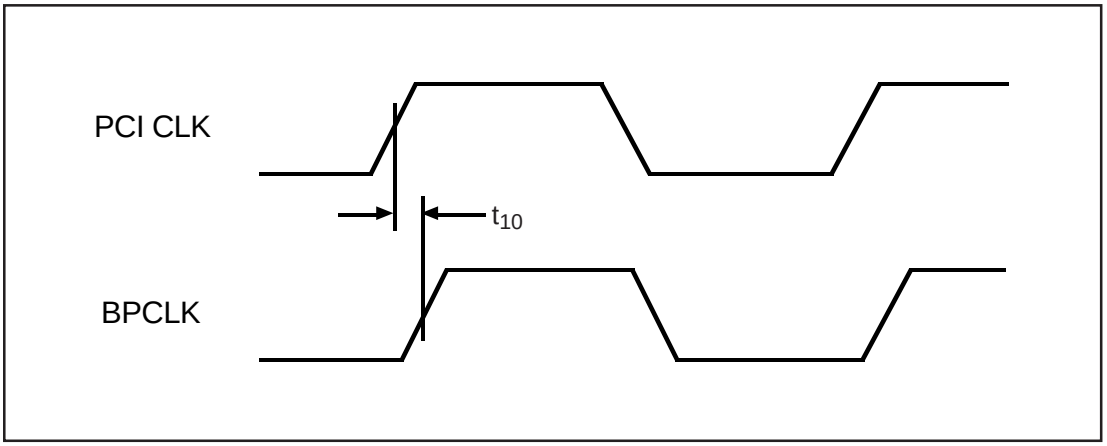


Figure 5. Pass-Thru Clock Relationship to PCI Clock

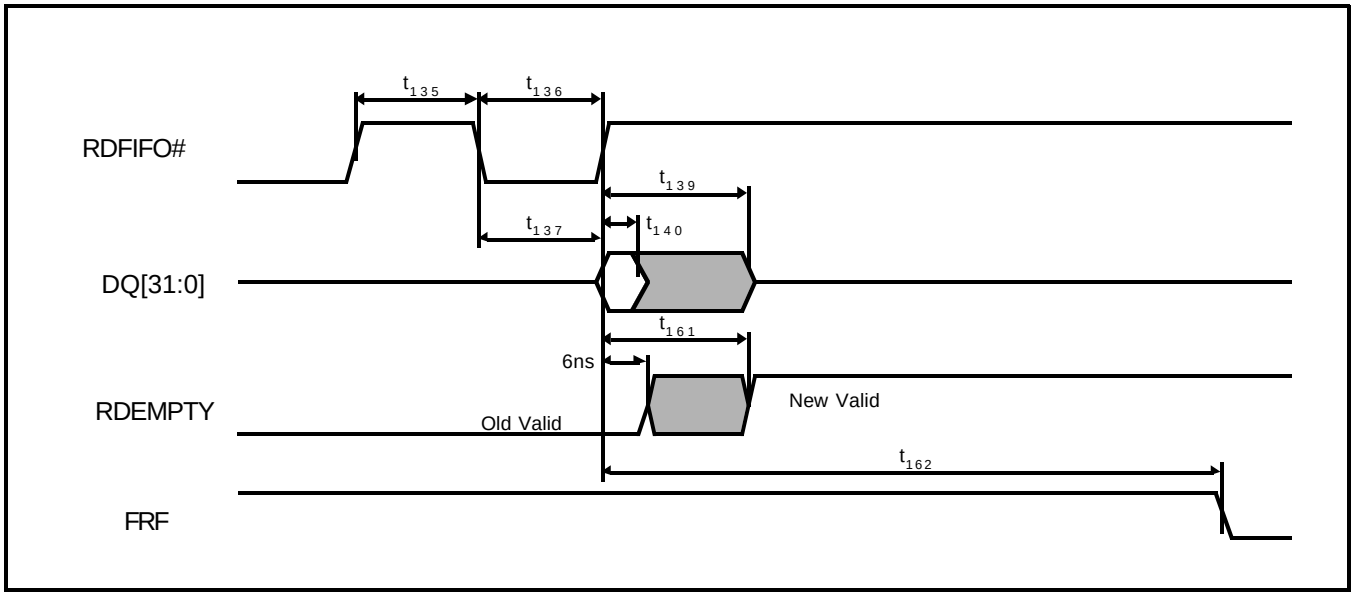


Asynchronous RDFIFO# Timing

Functional Operation Range (V_{CC}= 5.0V 5%, 0˚C to 70˚C T_a, 50 pf load on outputs).

Symbol	Parameter	Min	Max	Units	Notes
t_{135}	RDFIFO# High Time	17		ns	
t_{136}	RDFIFO# Low Time	17		ns	
t_{137}	RDFIFO# Low to DQ[31:0] Driven		21	ns	
t_{139}	RDFIFO# High to DQ[31:0] Float		20	ns	
t_{140}	DQ[31:0] Hold from RDFIFO# Rising Edge	5		ns	
t_{161}	PCI to ADD-ON FIFO RDEMPTY Valid from RDFIFO# Rising Edge		15	ns	
t_{162}	PCI to ADD-ON FIFO FRF Valid from RDFIFO# Rising Edge		85	ns	

Figure 6. Asynchronous RDFIFO# Timing

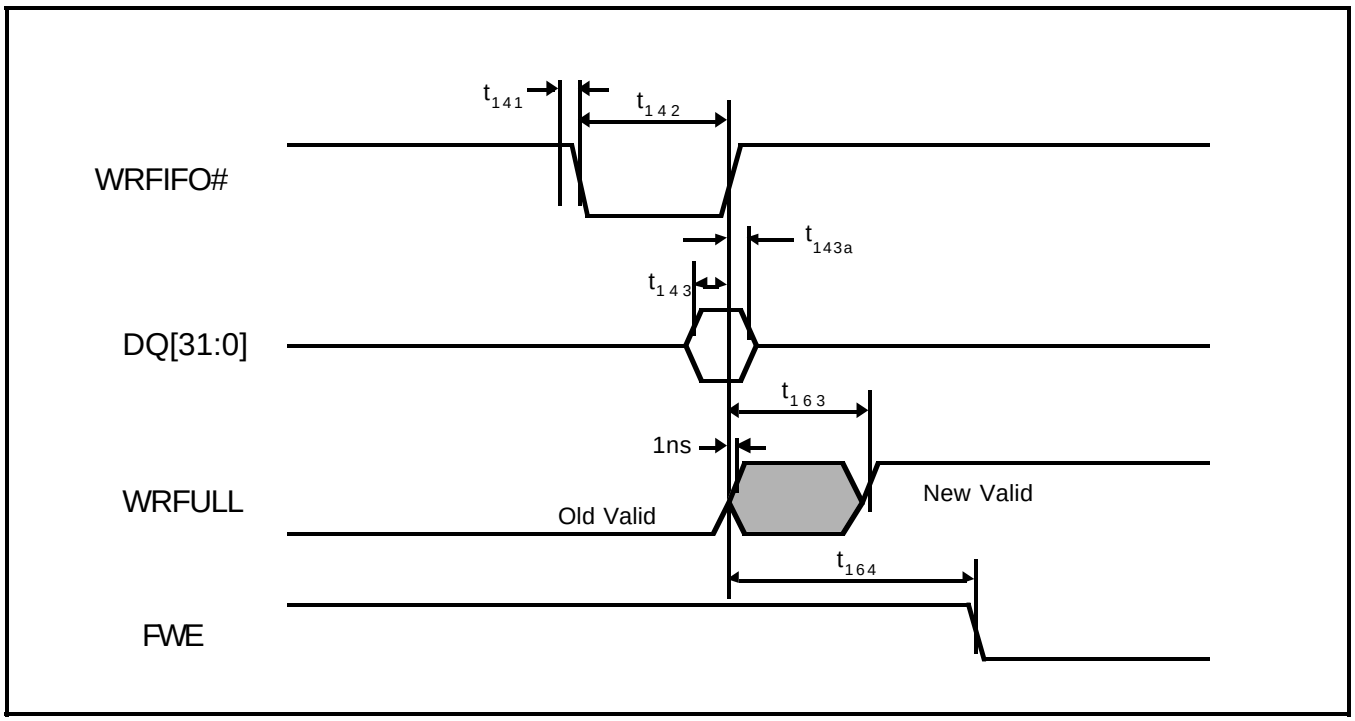


Asynchronous WRFIFO# Timing

Functional Operation Range (V_{CC}=5.0V 5%, 0˚C to 70˚C T_a, 50 pf load on outputs).

Symbol	Parameter	Min	Max	Units	Notes
t_{141}	WRFIFO# High Time	2		ns	
t_{142}	WRFIFO# Low Time	17		ns	
t_{143}	DQ[31:0] Setup to WRFIFO# Rising Edge	4		ns	
t_{143a}	DQ[31:0] Hold from WRFIFO# Rising Edge	2		ns	
t_{163}	ADD-ON to PCI FIFO WRFULL Valid from WRFIFO# Rising Edge		16	ns	
t_{164}	ADD-ON to PCI FIFO FWE Valid from WRFIFO# Rising Edge		28	ns	

Figure 7. Asynchronous WRFIFO# Timing



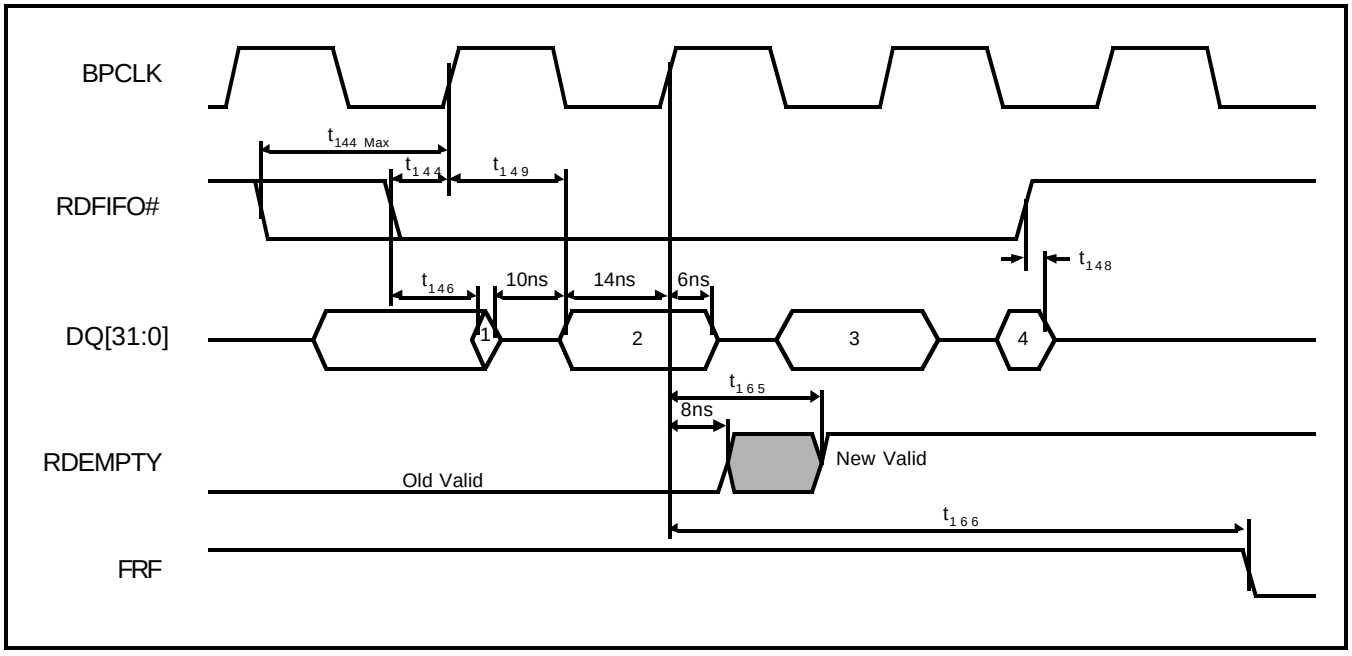
Synchronous RDFIFO# Timing

Functional Operation Range (V_{CC}=5.0V 5%, 0˚C to 70˚C T_a, 50 pf loaf on outputs).

Symbol	Parameter	Min	Max	Units	Notes
t_{144}	RDFIFO# Setup tp BPCLK Rising Edge	8	26	ns	1
t_{145}	RDFIFO# Low Time	8		ns	
t_{146}	RDFIFO# Low to DQ[31:0] Driven		12	ns	
t_{148}	RDFIFO# High to DQ[31:0] Float		3	ns	
t_{149}	DQ[31:0] Valid from BPCLK Rising Edge		16	ns	3
t_{165}	PCI to ADD-ON FIFO RDEMPTY Valid from BPCLK Rising Edge		12	ns	2
t_{166}	PCI to ADD-ON FIFO FRF Valid from BPCLK Rising Edge		80	ns	

Notes:
1. Min and Max times are indicated to allow increased valid data time as shown by dashed lines.
2. State change of RDEMPTY shown below is reference only. Actual change would indicate no Data 3 available.
3. Valid applies after first access. First access is async with following as sync accesses.

Figure 8. Synchronous RDFIFO# Timing



Synchronous WRFIFO# Timing

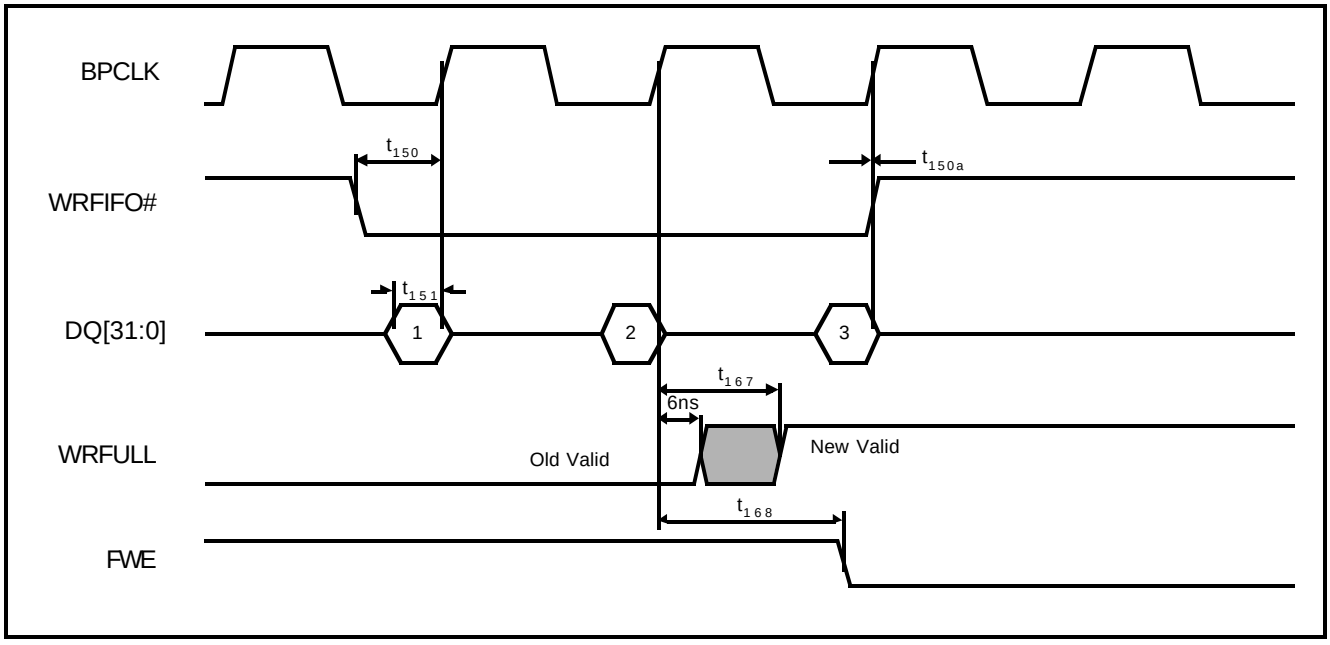
Functional Operation Range (V_{CC}= 5.0V 5%, 0˚C to 70˚C T_a, 50 pf load on outputs).

Symbol	Parameter	Min	Max	Units	Notes
t_{150}	WRFIFO# Setup to BPCLK Rising Edge	12		ns	
t_{150a}	WRFIFO# Hold Time to BPCLK Rising Edge		0	ns	
t_{151}	DQ[31:0] Setup to BPCLK Rising Edge	7			
t_{151a}	DQ[31:0] Hold from BPCLK Rising Edge		0		
t_{167}	ADD-ON to PCI WRFULL Valid from BPCLK Rising Edge		11	ns	1
t_{168}	ADD-ON to PCI FIFO FWE Valid from BPCLK Rising Edge		26	ns	

Notes:

1. State change of WRFULL shown below is reference only. Actual change would indicate no Data 3 written.

Figure 9. Synchronous WRFIFO# Timing

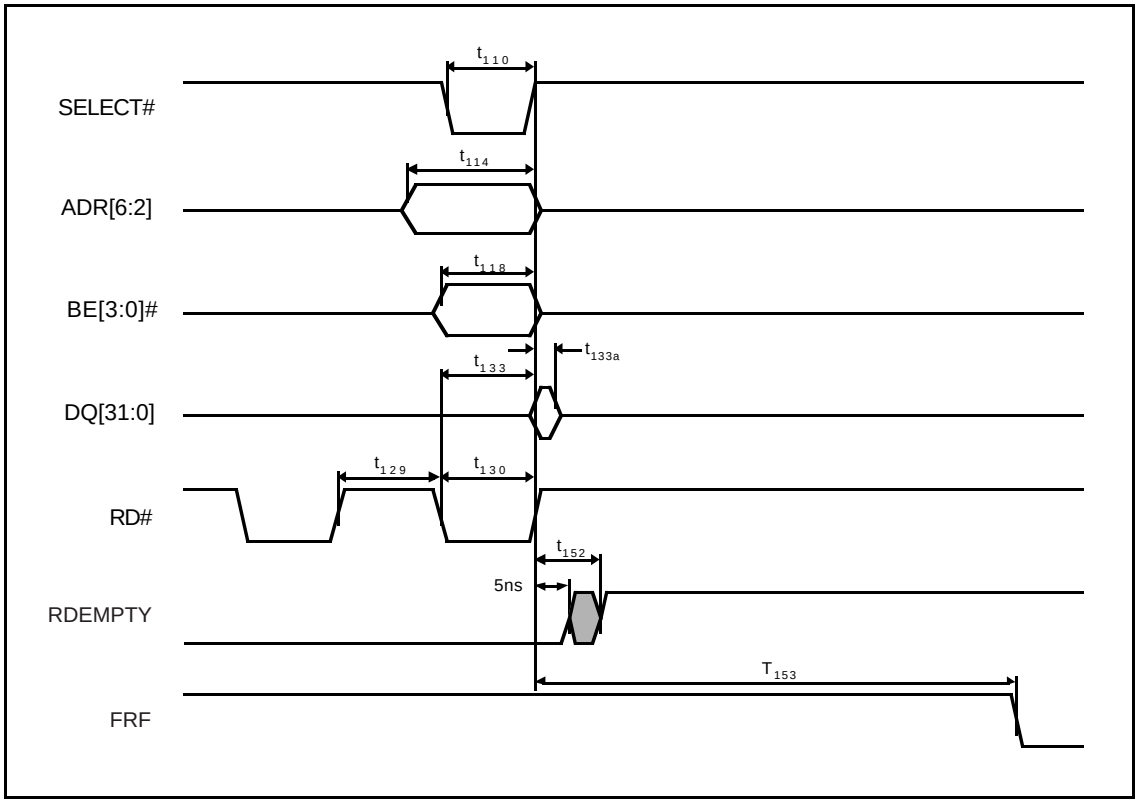


Asynchronous RD# FIFO and Register Access Timing

Functional Operation Range (V_{CC}=5.0V 5%, 0˚C to 70˚C T_a, 50 pf load on outputs).

Symbol	Parameter	Min	Max	Units	Notes
t_{110}	SELECT# Setup to RD# Rising Edge	10		ns	
t_{114a}	SELECT# Hold from RD# Rising Edge	-1		ns	
t_{114}	ADR[6:2] Setup to RD# Rising Edge	18		ns	
t_{114a}	ADR[6:2] Hold from RD# Rising Edge	0			
t_{118}	BE[3:0]# Setup to RD# Rising Edge	12		ns	
t_{118a}	BE[3:0]# Hold from RD# Rising Edge	0		ns	
t_{129}	RD# High Time	16		ns	
t_{130}	RD# Low Time	15		ns	
t_{133}	DQ[31:0] Valid from RD# Falling Edge	15		ns	
t_{133a}	DQ[31:0] Hold from RD# Rising Edge	3		ns	
t_{152}	RDEMPTY Status Valid from RD# Rising Edge		10	ns	
t_{153}	FRF Status Valid from RD# Rising Edge		75	ns	

Figure 10. Asynchronous RD# FIFO Timing

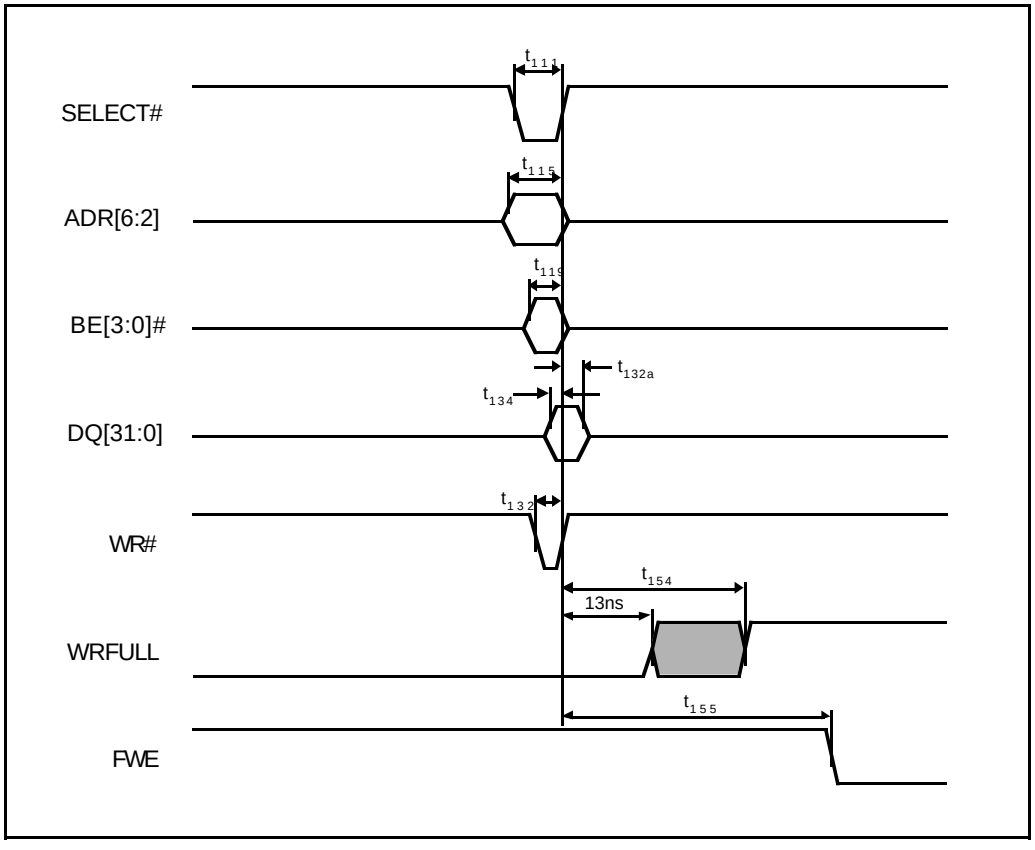


Asynchronous WR# FIFO and Register Access Timing

Functional Operation Range (V_{CC}=5.0V 5%, 0˚C to 70˚C T 50 pf load on outputs).

Symbol	Parameter	Min	Max	Units	Notes
t_{111}	SELECT# Setup to WR# Rising Edge	7		ns	
t_{111a}	SELECT# Hold from WR# Rising Edge	0		ns	
t_{115}	ADR[6:2] Setup to WR# Rising Edge	8		ns	
t_{115a}	ADR[6:2] Hold from WR# Rising Edge	0		ns	
t_{119}	BE[3:0]# Setup to WR# Rising Edge	5		ns	
t_{119a}	BE[3:0]# Hold from WR# Rising Edge	0		ns	
t_{131}	WR# High Time	TBD		ns	
t_{132}	WR# Low Time	4		ns	
t_{134}	DQ[31:0] Setup to WR# Rising Edge	2		ns	
t_{134a}	DQ[31:0] Hold from WR# Rising Edge	3		ns	
t_{154}	WRFULL Status Valid from WR# Rising Edge		27	ns	
t_{155}	FWE Status Valid from WR# Rising Edge		40	ns	

Figure 11. Asynchronous WR# FIFO Timing



Synchronous RD# FIFO Timing

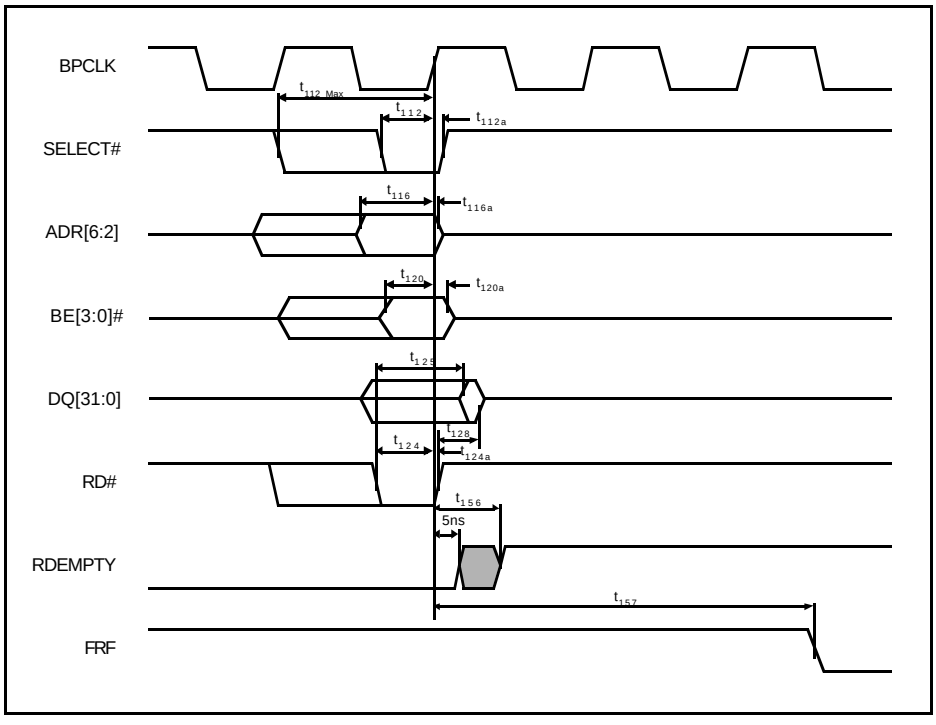
Functional Operation Range (V_{CC}=5.0V 5%, 0˚C to 70˚C T 50 pf load on outputs).

Symbol	Parameter	Min	Max	Units	Notes
t_{112}	SELECT# Setup to BPCLK Rising Edge	10	30	ns	4
t_{112a}	SELECT# Hold from BPCLK Rising Edge	2		ns	
t_{116}	ADR[6:2] Setup to BPCLK Rising Edge	14	34	ns	4
t_{116a}	ADR[6:2] Hold from BPCLK Rising Edge	1		ns	
t_{120}	BE[3:0]# Setup to BPCLK Rising Edge	9	29	ns	4
t_{120a}	BE[3:0]# Hold from BPCLK Rising Edge	3		ns	
t_{125}	RD# Low to DQ[31:0] Driven		17	ns	1
t_{128}	RD# High to DQ[31:0] Float		8	ns	
t_{156}	RDEMPTY Status Valid to BPCLK Rising Edge		13	ns	
t_{157}	FRF Status Valid to BPCLK Rising Edge		74	ns	
t_{124}	RD# Setup to BPCLK Rising Edge	11	31	ns	4
t_{124a}	RD# Hold from BPCLK Rising Edge	1		ns	
t_{127}	DQ[31:0] Valid from BPCLK Rising Edge		6	ns	

Notes:

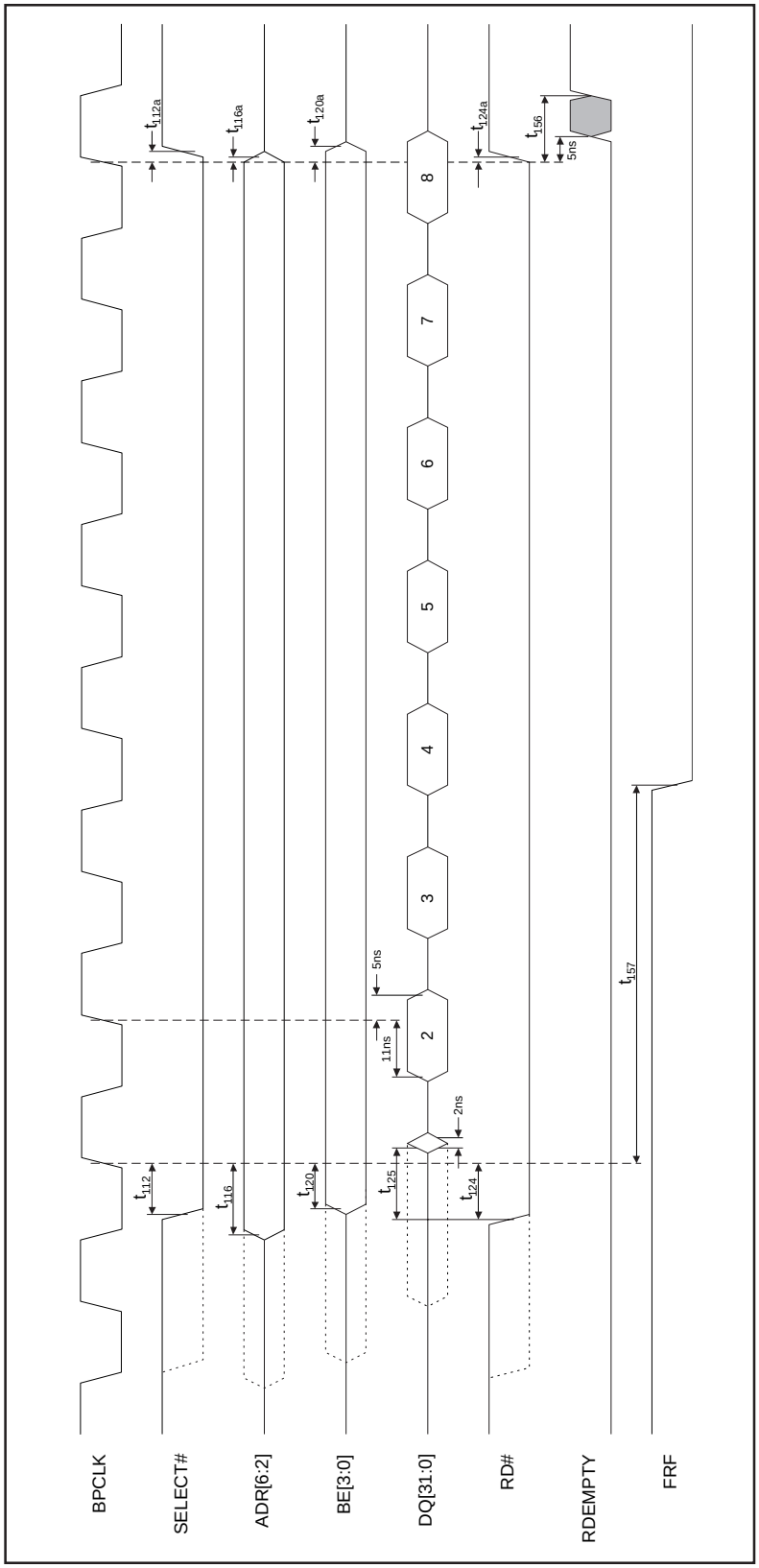
1. Data is valid for 22ns for a 31ns t_{124} RD# Setup.
2. RD# and SELECT# must both be asserted to dric=ve DQ[31:0] - delay is from the last one asserted.
3. When increasing Setup times, ADR[6:2], BE[3:0]#, SELECT#, and RD# timing relations remain relative to each other as shown.
4. Min and Max are indicated to allow increased valid data time as shown by dashed lines. First accesses are async.

Figure 12. Synchronous RD# FIFO Timing



Synchronous Multiple RD# FIFO Timing

Figure 13. Synchronous RD# FIFO Timing

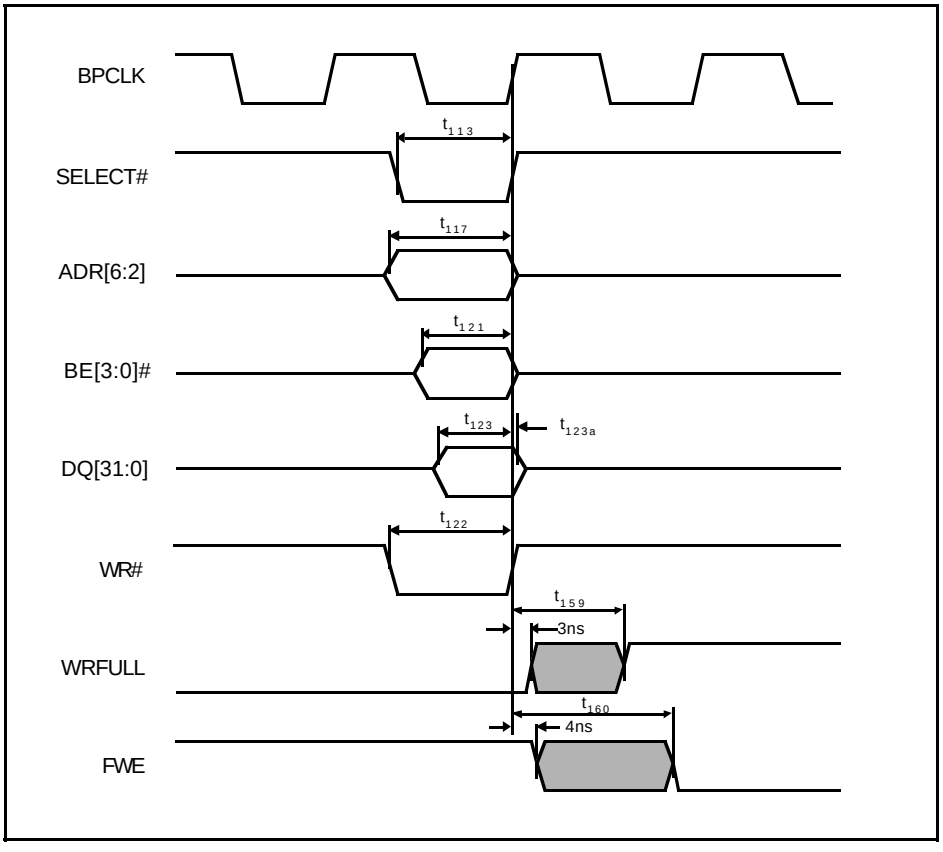


Synchronous WR# FIFO Timing

Functional Operation Range (V_{CC}=5.0V 5%, 0˚C to 70˚C T_a, 50 pf load on outputs).

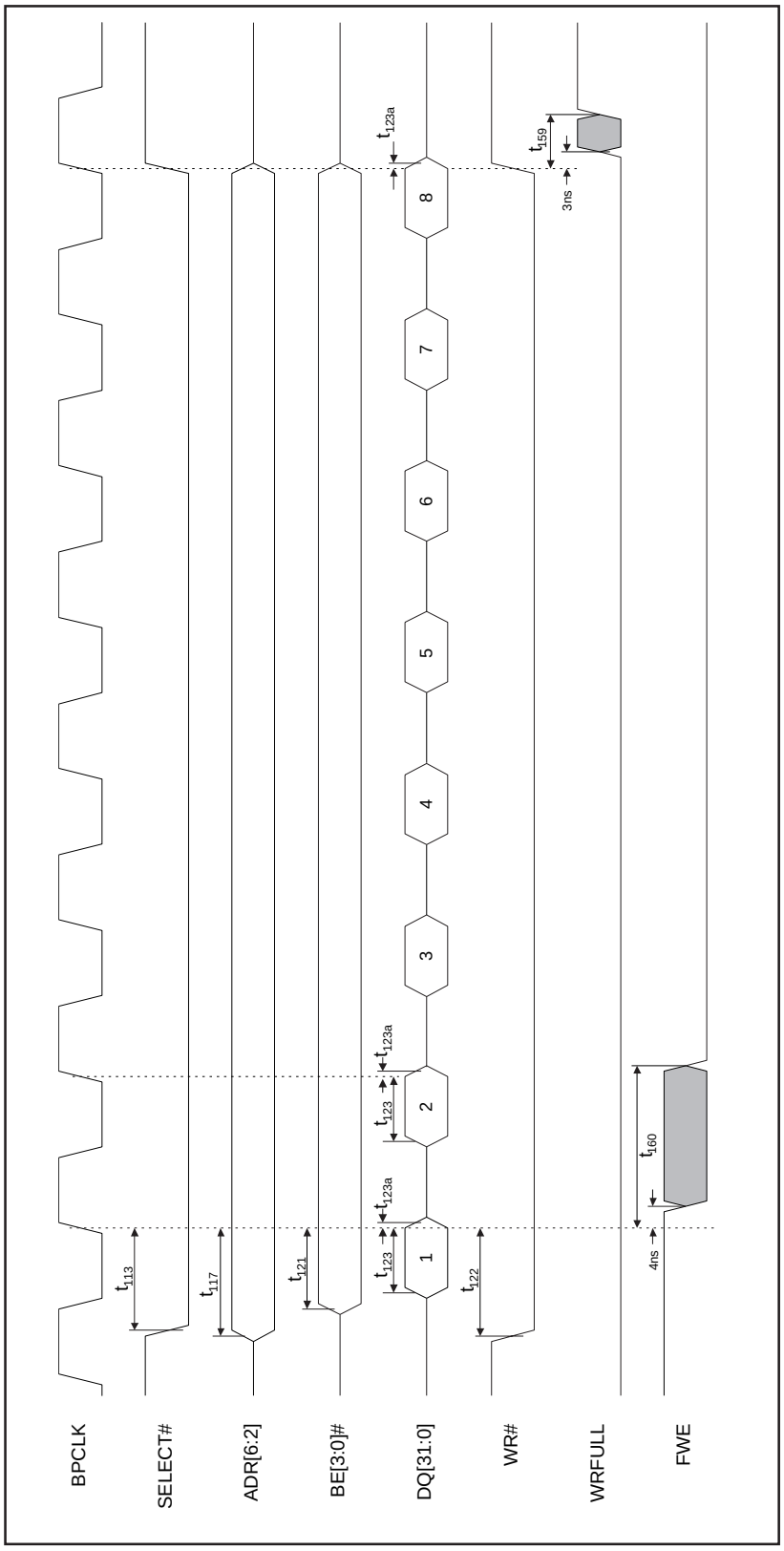
Symbol	Parameter	Min	Max	Units	Notes
t_{113}	SELECT# Setup to BPCLK Rising Edge	19		ns	
t_{113a}	SELECT# Hold from BPCLK Rising Edge	0		ns	
t_{117}	ADR[6:2] Setup to BPCLK Rising Edge	20		ns	
t_{117a}	ADR[6:2] Hold from BPCLK Rising Edge	0		ns	
t_{121}	BE[3:0]# Setup to BPCLK Rising Edge	15		ns	
t_{121a}	BE[3:0]# Hold from BPCLK Rising Edge	0		ns	
t_{123}	DQ[31:0] Setup to BPCLK Rising Edge	12		ns	
t_{123a}	DQ[31:0] Hold from BPCLK Rising Edge	1		ns	
t_{122}	WR# Setup to BPCLK Rising Edge	20		ns	
t_{122a}	WR# Hold from BPCLK Rising Edge	0		ns	
t_{159}	WRFULL Status Valid to BPCLK Rising Edge		18	ns	
t_{160}	FWE Status Valid to BPCLK Rising Edge		26	ns	

Figure 14. Synchronous WR# FIFO Timing



Synchronous Multiple WR# FIFO Timing

Figure 15. Synchronous Multiple WR# FIFO Timing



Target S5933 Pass-Thru Interface Timings

Functional Operation Range (V_{CC}=5.0V ±5%, 0˚C to 70˚C, 50 pF load on outputs)

Symbol	Parameter	Min	Max	Units	Notes
t_{10a}	SELECT# Setup to BPCLK Rising Edge	3		ns	
t_{11a}	SELECT# Hold from BPCLK Rising Edge	2		ns	
t_{13}	ADR[6:2], BE[3:0]# Setup to BPCLK Rising Edge	5		ns	
t_{14}	ADR[6:2], BE[3:0]# Hold from BPCLK Rising Edge	2		ns	
t_{17}	RD# Low to DQ{31:0] Driven		13	ns	1
t_{24}	Pass-Thru Status Valid from BPCLK Rising Edge		5	ns	
t_{25}	Pass-Thru Status Hold from BPCLK Rising Edge	0		ns	
t_{26}	PTRDY# Setup to BPCLK Rising Edge	5		ns	
t_{27}	PTRDY# Hold from BPCLK Rising Edge	3		ns	
t_{28}	PCICLK to BPCLK delay	2	6.5	ns	
t_{29}	RD#, WR# Setup to BPCLK Rising Edge	5		ns	
t_{30}	RD#, WR# Hold from BPCLK Rising Edge	2		ns	
t_{31}	DQ[31:0] Setup to BPCLK Rising Edge	5		ns	
t_{32}	DQ[31:0] Hold from BPCLK Rising Edge	2		ns	
t_{33}	DQ[31:0] Valid from BPCLK Rising Edge		15	ns	
t_{34}	DQ[31:0] Float from RD# Rising Edge		12	ns	

Notes:

1. This timing also applies to the use of BE[3:0]# to control DQ[31:0] drive.

Figure 17. Pass-Thru Data Register Read Timing

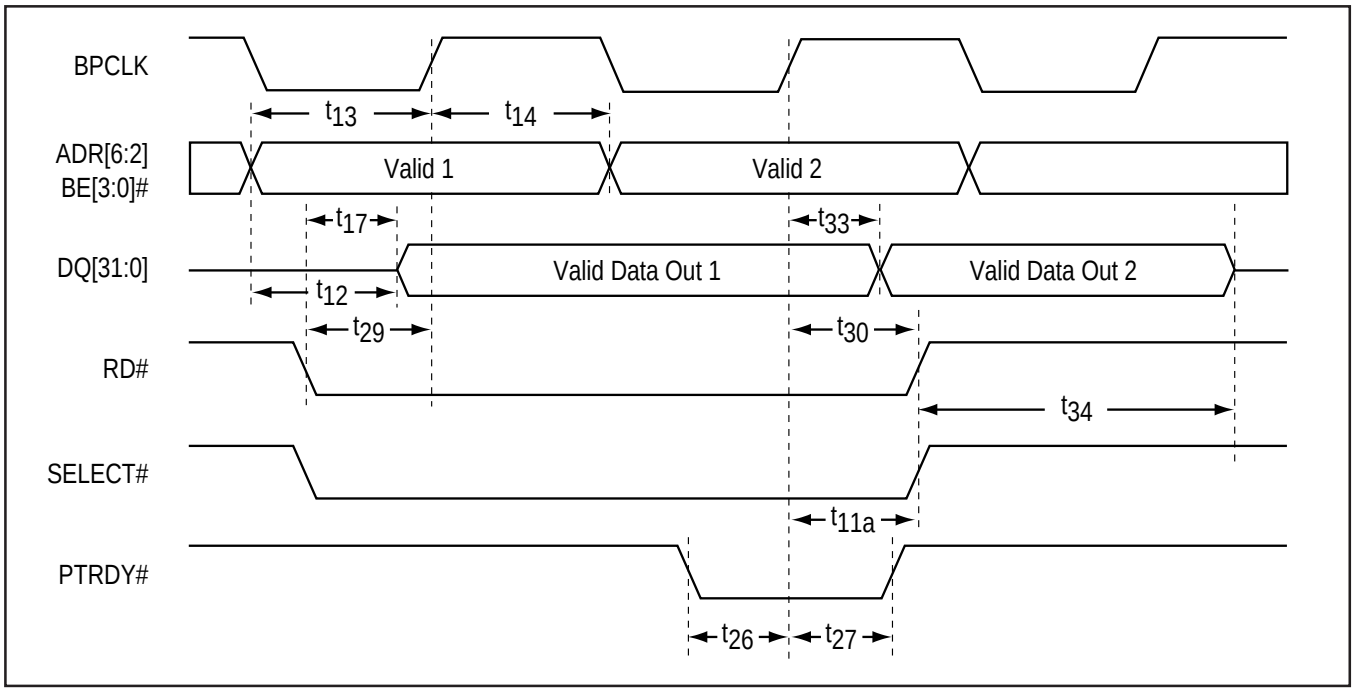


Figure 18. Pass-Thru Data Register Write Timing

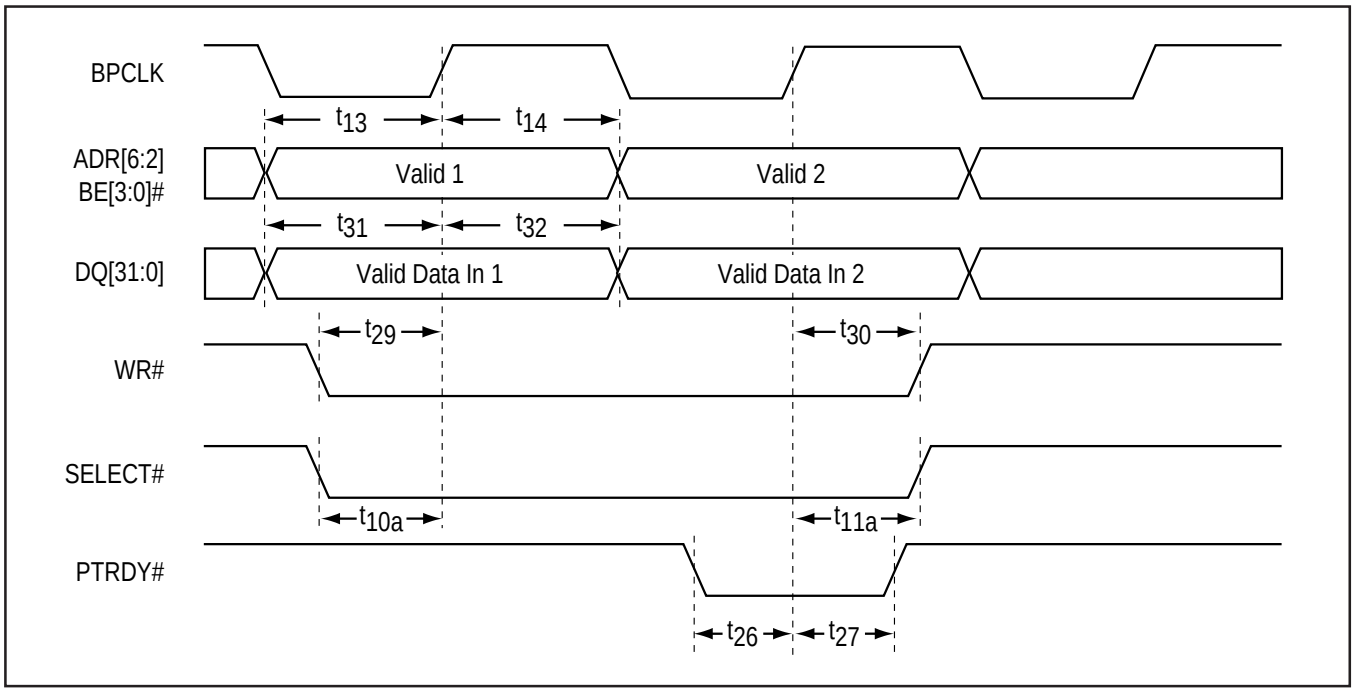
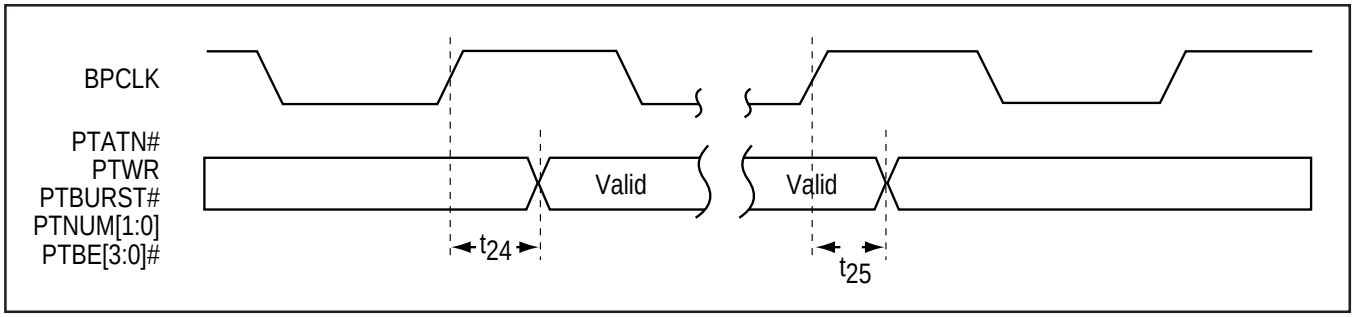


Figure 19. Pass-Thru Status Indicator Timing



Target Byte-Wide nv Memory Interface Timings

Functional Operation Range (V_{CC}=5.0V ±5%, 0˚C to 70˚C, 50 pF load on outputs)

Symbol	Parameter	Min	Max	Units	Notes
t35	ERD# Cycle Time	8T		ns	Note 1
t36	ERD# Low Time	6T		ns	Note 1
t37	ERD# High Time	2T		ns	Note 1
t38	EA[15:0] Setup to ERD# or EWR# Low	T		ns	Note 1
t39	EA[15:0] Hold from ERD# or EWR# High	T		ns	Note 1
t40	EQ[7:0] Setup to ERD# Rising Edge	10		ns	Note 1
t41	EQ[7:0] Hold from ERD# Rising Edge	2		ns	Note 1
t42	EWR# Cycle Time			ns	Note 1,2
t43	EWR# Low Time	6T		ns	Note 1
t44	EWR# High Time	2T		ns	Note 1
t45	EQ[7:0] Setup to EWR# Low	-10	0	ns	Note 1
t46	EQ[7:0] Hold from EWR# High	T		ns	Note 1

Notes:

1. T represents the clock period for the PCI bus clock (30ns @ 33 MHz).
2. The write cycle time is controlled by both the PCI bus clock and software operations to initiate the write operation of nv memory. This parameter is the result of several software operations to the Bus Master Control/Status Register (MCSR).

Figure 20. nv Memory Read Timing

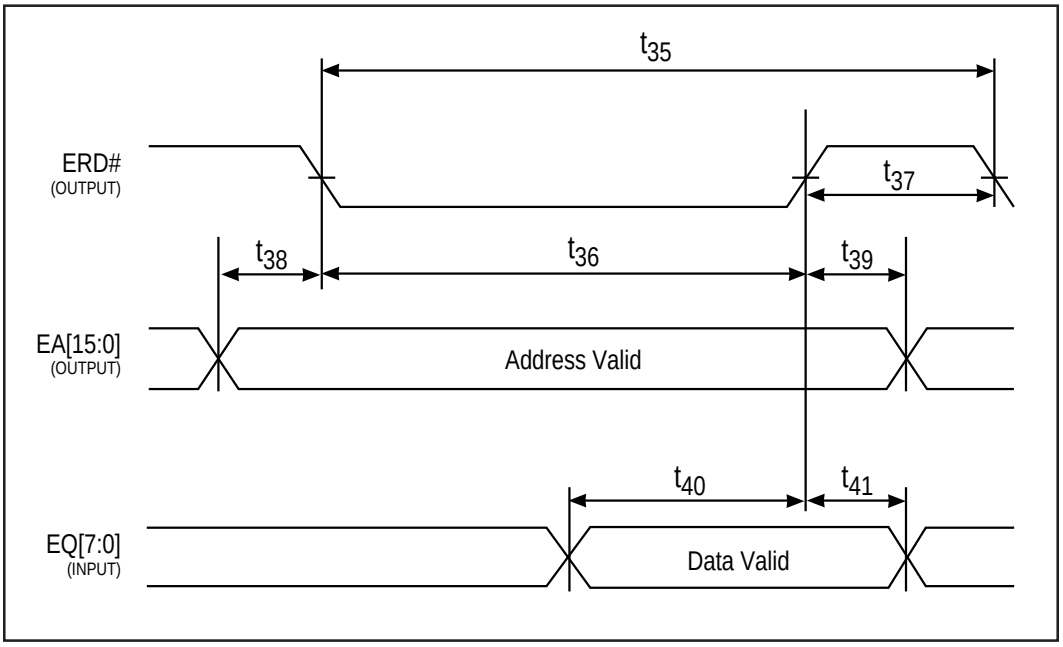
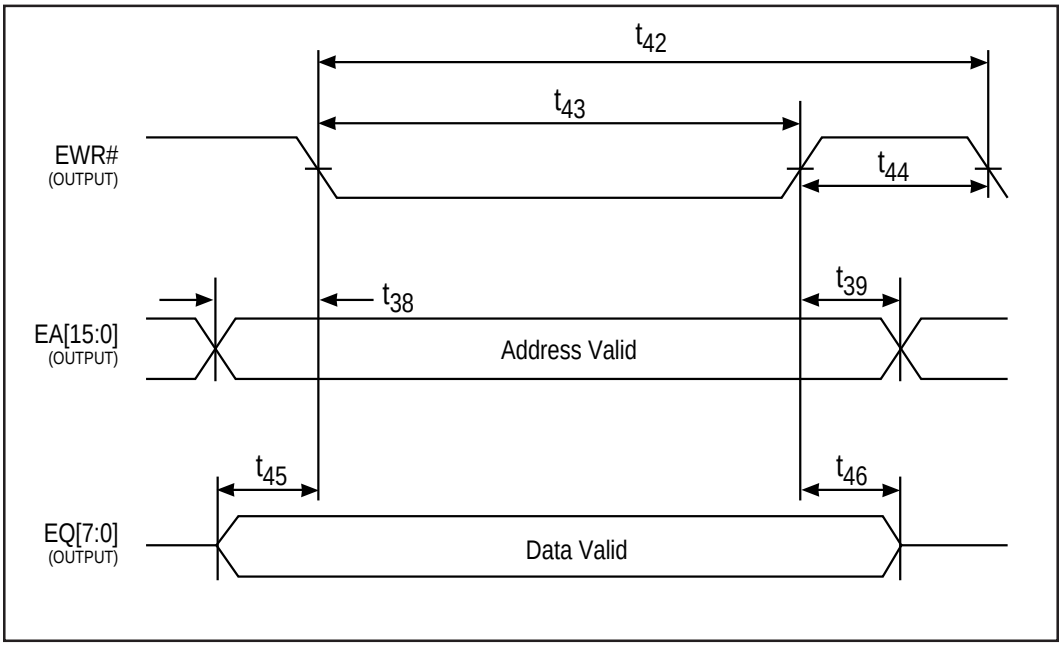


Figure 21. nv Memory Write Timing



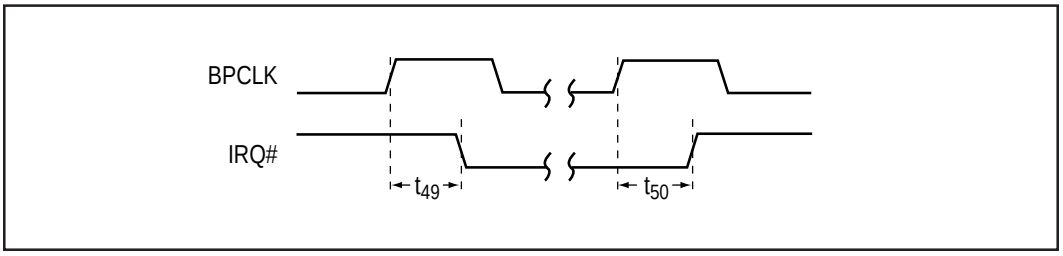
Target Interrupt Timings

Functional Operation Range (V_{CC}=5.0V ±5%, 0˚C to 70˚C, 50 pF load on outputs)

Symbol	Parameter	Min	Max	Units	Notes
t49	IRQ# Low from BPCLK Rising Edge		15	ns	Note 1
t50	IRQ# High from BPCLK Rising Edge		15	ns	Note 1

Notes:
1. This timing applies to interrupts generated and cleared from the PCI interface.

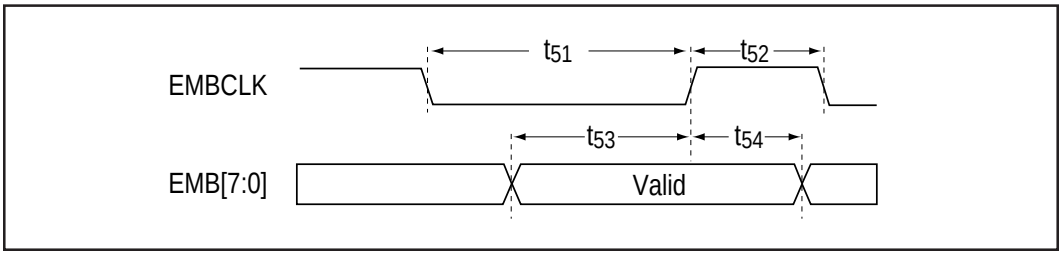
Figure 22. IRQ# Interrupt Output Timing



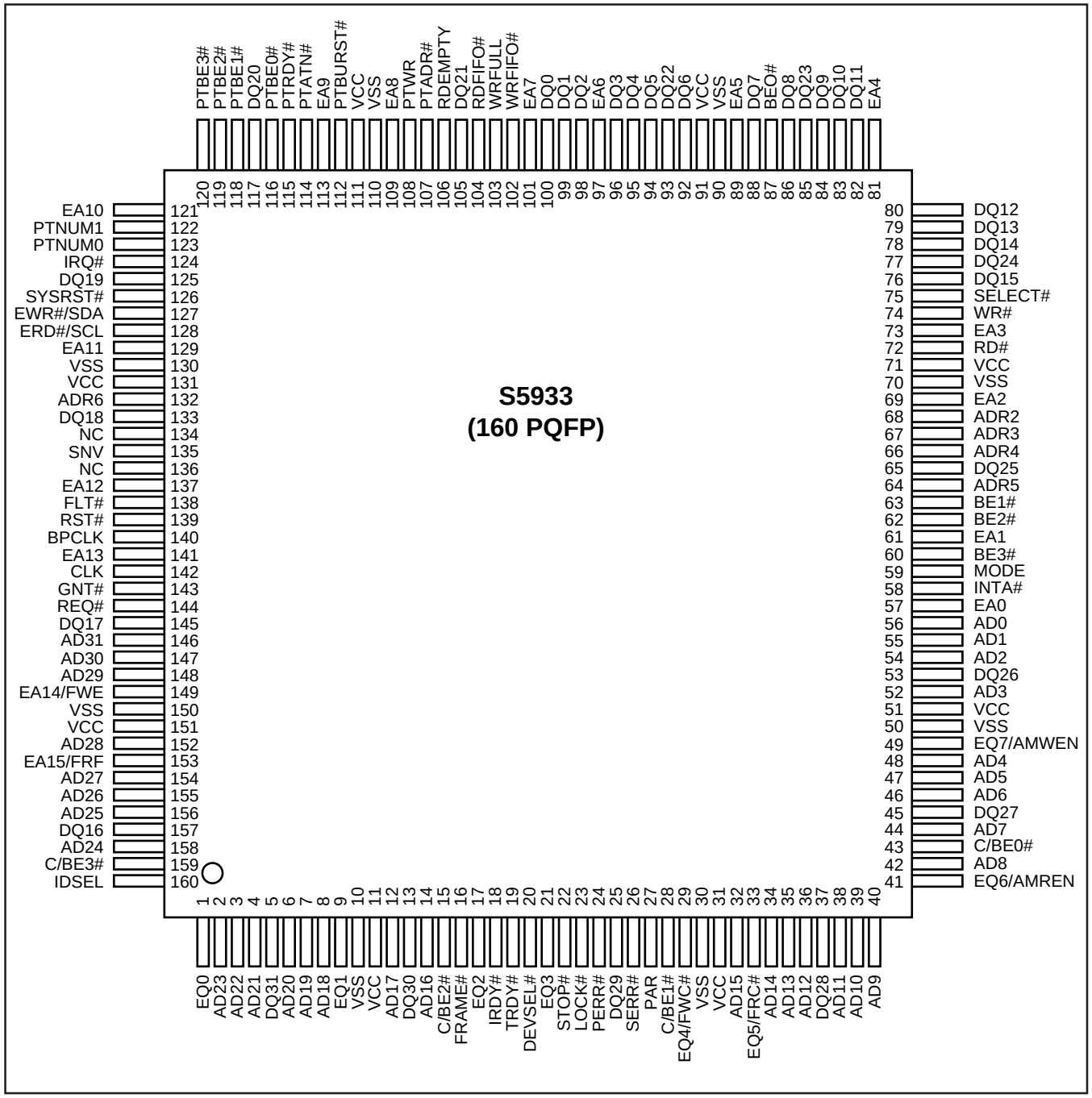
Functional Operation Range (V_{CC}=5.0V ±5%, 0˚C to 70˚C, 50 pF load on outputs)

Symbol	Parameter	Min	Max	Units	Notes
t51	EMBCLK Low Time	12		ns	
t52	EMBLK High Time	12		ns	
t53	EMB[7:0] Setup to EMBCLK Rising Edge	5		ns	
t54	EMB[7:0] Hold from EMBCLK Rising Edge	2		ns	

Figure 23. Mailbox 4, Byte 3 Direct Input Timing



S5933 Pinout and Pin Assignment - 160 PQFP



S5933 Pinout and Pin Assignment - 208 TQFP

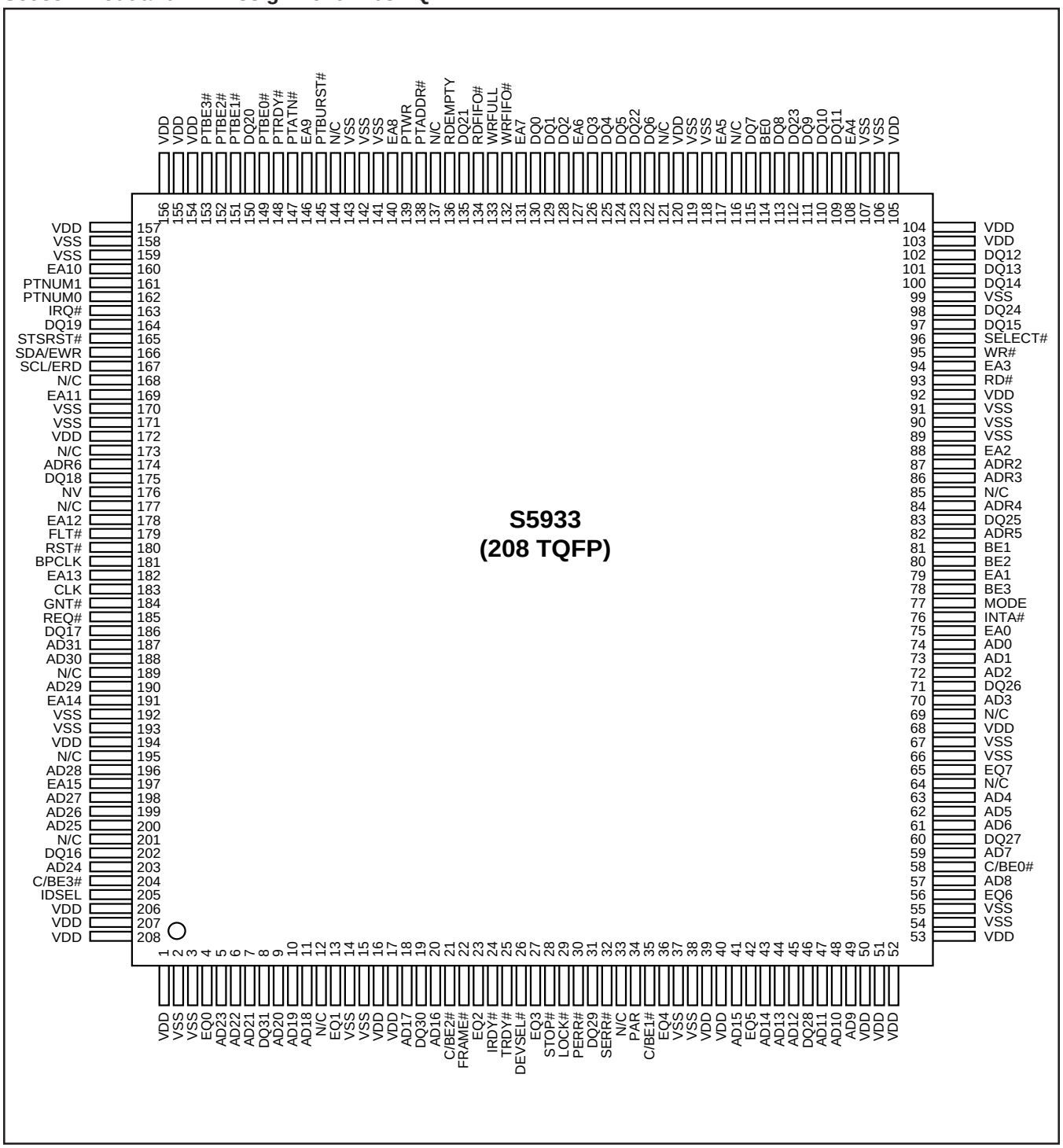


Table 1. S5933 Numerical Pin Assignment - 160 PQFP

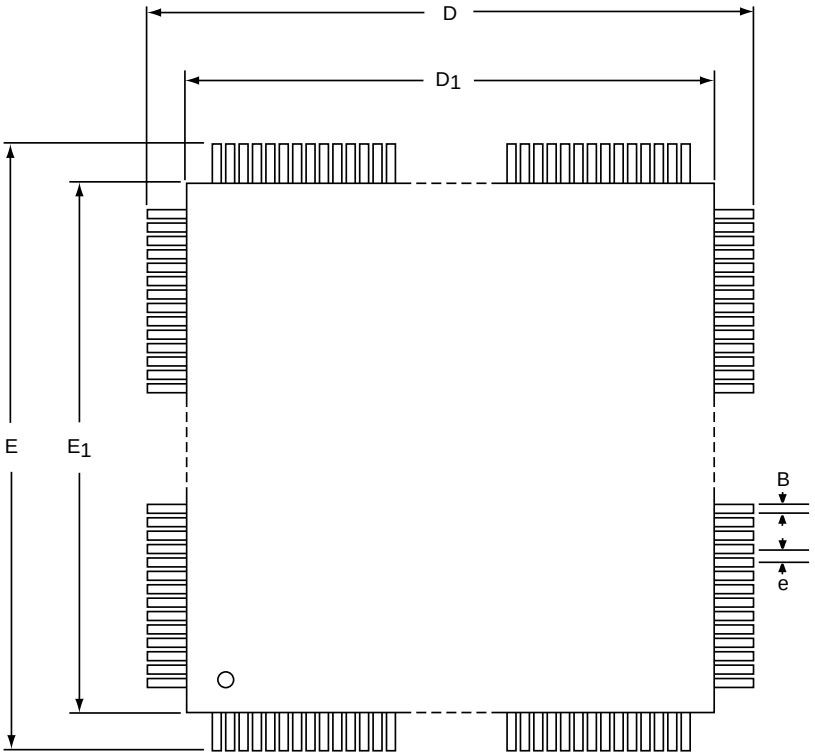
Pin#	Signal	Type
1	EQ0	t/s
2	AD23	t/s
3	AD22	t/s
4	AD21	t/s
5	DQ31	t/s
6	AD20	t/s
7	AD19	t/s
8	AD18	t/s
9	EQ1	t/s
10	VSS	V
11	VCC	V
12	AD17	t/s
13	DQ30	t/s
14	AD16	t/s
15	C/BE2#	t/s
16	FRAME#	t/s
17	EQ2	t/s
18	IRDY#	t/s
19	TRDY#	t/s
20	DEVSEL#	t/s
21	EQ3	t/s
22	STOP#	t/s
23	LOCK#	in
24	PERR#	t/s
25	DQ29	t/s
26	SERR#	o/d
27	PAR	t/s
28	C/BE1#	t/s
29	EQ4/FWC#	t/s
30	VSS	V
31	VCC	V
32	AD15	t/s
33	EQ5/FRC#	t/s
34	AD14	t/s
35	AD13	t/s
36	AD12	t/s
37	DQ28	t/s
38	AD11	t/s
39	AD10	t/s
40	AD9	t/s
41	EQ6/AMREN	t/s
42	AD8	t/s
43	C/BE0#	t/s
44	AD7	t/s
45	DQ27	t/s
46	AD6	t/s
47	AD5	t/s
48	AD4	t/s
49	EQ7/AMWEN	t/s
50	VSS	V
51	VCC	V
52	AD3	t/s
53	DQ26	t/s
54	AD2	t/s
55	AD1	t/s
56	AD0	t/s
57	EA0	t/s
58	INTA#	o/d
59	MODE	in
60	BE3#	in
61	EA1	t/s
62	BE2#	in
63	BE1#	in
64	ADR5	in
65	DQ25	t/s
66	ADR4	in
67	ADR3	in
68	ADR2	in
69	EA2	t/s
70	VSS	V
71	VCC	V
72	RD#	in
73	EA3	t/s
74	WR#	in
75	SELECT#	in
76	DQ15	t/s
77	DQ24	t/s
78	DQ14	t/s
79	DQ13	t/s
80	DQ12	t/s
81	EA4	t/s
82	DQ11	t/s
83	DQ10	t/s
84	DQ9	t/s
85	DQ23	t/s
86	DQ8	t/s
87	BE0#	in
88	DQ7	t/s
89	EA5	t/s
90	VSS	V
91	VCC	V
92	DQ6	t/s
93	DQ22	t/s
94	DQ5	t/s
95	DQ4	t/s
96	DQ3	t/s

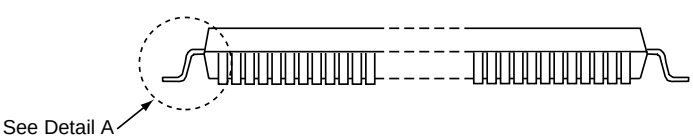
Table 1. S5933 Numerical Pin Assignment - 160 PQFP (Continued)

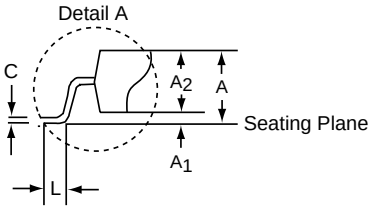
Pin#	Signal	Type
97	EA6	t/s
98	DQ2	t/s
99	DQ1	t/s
100	DQ0	t/s
101	EA7	t/s
102	WRFIFO#	in
103	WRFULL	out
104	RDFIFO#	in
105	DQ21	t/s
106	RDEMPTY	out
107	PTADR#	in
108	PTWR	out
109	EA8	t/s
110	VSS	V
111	VCC	V
112	PTBURST#	out
113	EA9	out
114	PTATN#	out
115	PTRDY#	in
116	PTBE0#	out
117	DQ20	t/s
118	PTBE1#	out
119	PTBE2#	out
120	PTBE3#	out
121	EA10	out
122	PTNUM1	out
123	PTNUM0	out
124	IRQ#	out
125	DQ19	t/s
126	SYSRST#	out
127	EWR#/SDA	t/s
128	ERD#/SCL	out

Pin#	Signal	Type
129	EA11	out
130	VSS	V
131	VCC	V
132	ADR6	in
133	DQ18	t/s
134	NC	—
135	SNV	in
136	NC	—
137	EA12	out
138	FLT#	in
139	RST#	in
140	BPCLK	out
141	EA13	out
142	CLK	in
143	GNT	in
144	REQ#	out
145	DQ17	t/s
146	AD31	t/s
147	AD30	t/s
148	AD29	t/s
149	EA14/FWE	t/s
150	VSS	V
151	VCC	V
152	AD28	t/s
153	EA15/FRF	t/s
154	AD27	t/s
155	AD26	t/s
156	AD25	t/s
157	DQ16	t/s
158	AD24	t/s
159	C/BE3#	t/s
160	IDSEL	in

Package Physical Dimensions - 160 PQFP







PQFP IN MILLIMETERS		
LEAD#	160	
SYMBOL	MIN	MAX
A	---	4.07
A1	0.25	---
A2	3.17	3.87
B	0.22	0.38
C	0.15	0.20
D1	27.90	28.10
E1	27.90	28.10
e	0.65	BSC
D	31.65	32.15
E	31.65	32.15
L	0.65	0.95

Table 2. S5933 Numerical Pin Assignment - 208 TQFP

Pin#	Signal	Type
1	VDD	V
2	VSS	V
3	VSS	V
4	EQ0	t/s
5	AD23	t/s
6	AD22	t/s
7	AD21	t/s
8	DQ31	t/s
9	AD20	t/s
10	AD19	t/s
11	AD18	t/s
12	N/C	---
13	EQ1	t/s
14	VSS	V
15	VSS	V
16	VDD	V
17	VDD	V
18	AD17	t/s
19	DQ30	t/s
20	AD16	t/s
21	C/BE2#	t/s
22	FRAME#	t/s
23	EQ2	t/s
24	IRDY#	t/s
25	TRDY#	t/s
26	DEVSEL#	t/s
27	EQ3	t/s
28	STOP#	t/s
29	LOCK#	I
30	PERR#	t/s
31	DQ29	t/s
32	SERR#	O

Pin#	Signal	Type
33	N/C	---
34	PAR	t/s
35	C/BE1#	t/s
36	EQ4	t/s
37	VSS	V
38	VSS	V
39	VDD	V
40	VDD	V
41	AD15	t/s
42	EQ5	t/s
43	AD14	t/s
44	AD13	t/s
45	AD12	t/s
46	DQ28	t/s
47	AD11	t/s
48	AD10	t/s
49	AD9	t/s
50	VDD	V
51	VDD	V
52	VDD	V
53	VDD	V
54	VSS	V
55	VSS	V
56	EQ6	t/s
57	AD8	t/s
58	C/BE0#	t/s
59	AD7	t/s
60	DQ27	t/s
61	AD6	t/s
62	AD5	t/s
63	AD4	t/s
64	N/C	---

Pin#	Signal	Type
65	EQ7	t/s
66	VSS	V
67	VSS	V
68	VDD	V
69	N/C	---
70	AD3	t/s
71	DQ26	t/s
72	AD2	t/s
73	AD1	t/s
74	AD0	t/s
75	EA0	t/s
76	INTA#	O
77	MODE	I
78	BE3	I
79	EA1	t/s
80	BE2	I
81	BE1	I
82	ADR5	I
83	DQ25	t/s
84	ADR4	I
85	N/C	---
86	ADR3	I
87	ADR2	I
88	EA2	t/s
89	VSS	V
90	VSS	V
91	VSS	V
92	VDD	V
93	RD#	I
94	EA3	t/s
95	WR#	I
96	SELECT#	I

Table 2. S5933 Numerical Pin Assignment - 208 TQFP (Continued)

Pin#	Signal	Type
97	DQ15	t/s
98	DQ24	t/s
99	VSS	V
100	DQ14	t/s
101	DQ13	t/s
102	DQ12	t/s
103	VDD	V
104	VDD	V
105	VDD	V
106	VSS	V
107	VSS	V
108	EA4	t/s
109	DQ11	t/s
110	DQ10	t/s
111	DQ9	t/s
112	DQ23	t/s
113	DQ8	t/s
114	BE0	I
115	DQ7	t/s
116	N/C	---
117	EA5	t/s
118	VSS	V
119	VSS	V
120	VDD	V
121	N/C	---
122	DQ6	t/s
123	DQ22	t/s
124	DQ5	t/s
125	DQ4	t/s
126	DQ3	t/s
127	EA6	t/s
128	DQ2	t/s

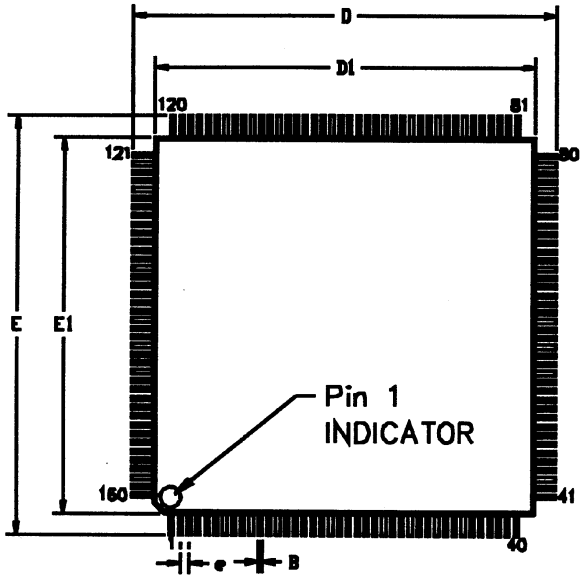
Pin#	Signal	Type
129	DQ1	t/s
130	DQ0	t/s
131	EA7	t/s
132	WRFIFO#	I
133	WRFULL	O
134	RDFIFO#	I
135	DQ21	t/s
136	RDEMPTY	O
137	N/C	---
138	PTADDR#	I
139	PTWR	O
140	EA8	t/s
141	VSS	V
142	VSS	V
143	VSS	V
144	N/C	---
145	PTBURST#	O
146	EA9	t/s
147	PTATN#	O
148	PTRDY#	I
149	PTBE0#	O
150	DQ20	t/s
151	PTBE1#	O
152	PTBE2#	O
153	PTBE3#	O
154	VDD	V
155	VDD	V
156	VDD	V
157	VDD	V
158	VSS	V
159	VSS	V
160	EA10	O

Pin#	Signal	Type
161	PTNUM1	O
162	PTNUM0	O
163	IRQ#	O
164	DQ19	t/s
165	SYSRST#	O
166	SDA/EWR	O
167	SCL/ERD	O
168	N/C	---
169	EA11	O
170	VSS	V
171	VSS	V
172	VDD	V
173	N/C	---
174	ADR6	I
175	DQ18	t/s
176	NV	I
177	N/C	---
178	EA12	O
179	FLT#	I
180	RST#	I
181	BPCLK	t/s
182	EA13	O
183	CLK	I
184	GNT#	I
185	REQ#	O
186	DQ17	t/s
187	AD31	t/s
188	AD30	t/s
189	N/C	---
190	AD29	t/s
191	EA14	O
192	VSS	V

Table 2. S5933 Numerical Pin Assignment - 208 TQFP (Continued)

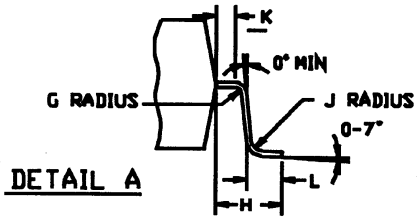
Pin#	Signal	Type
193	VSS	V
194	VDD	V
195	N/C	---
196	AD28	t/s
197	EA15	O
198	AD27	t/s
199	AD26	t/s
200	AD25	t/s
201	N/C	---
202	DQ16	t/s
203	AD24	t/s
204	C/BE3#	t/s
205	IDSEL	I
206	VDD	V
207	VDD	V
208	VDD	V

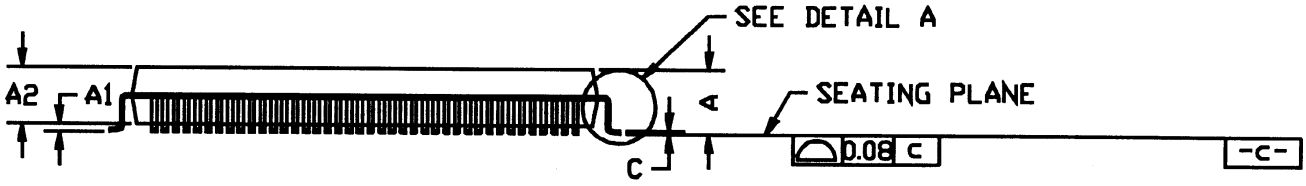
Package Physical Dimensions - 208 TQFP



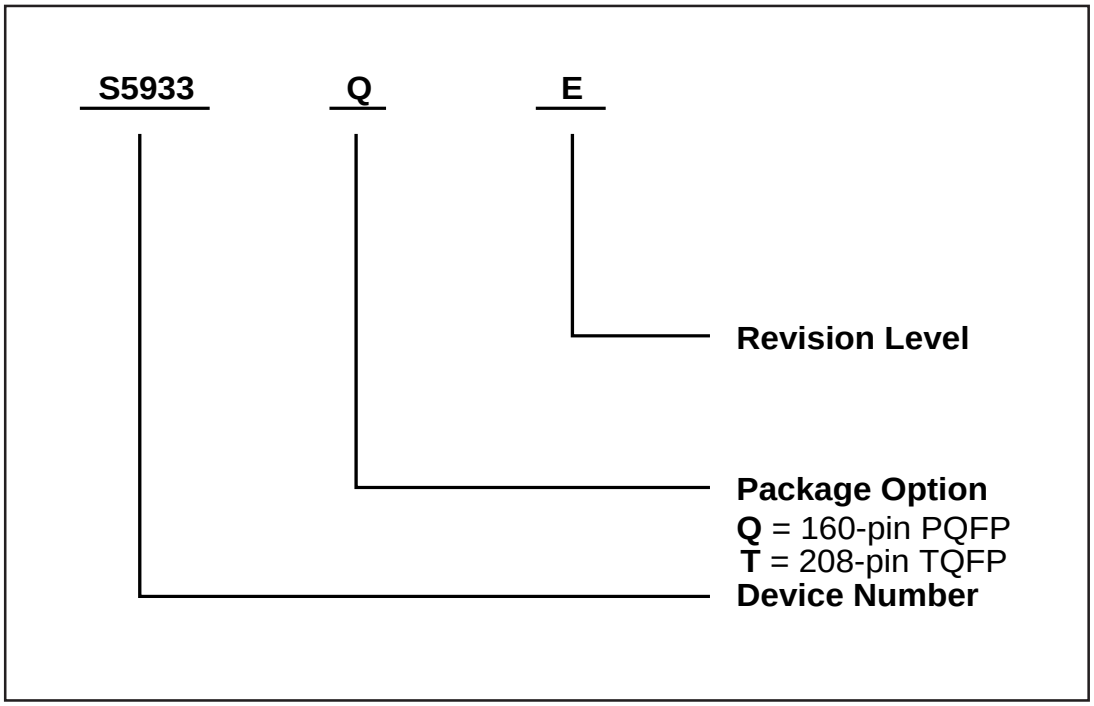
Symbol	MIN	NOM	MAX
A	-	-	1.60
A1	0.05	-	0.15
A2	1.35	1.40	1.45
D		30.00 BSC	
D1		28.00 BSC	
E		30.00 BSC	
E1		28.00 BSC	
L	0.45	0.60	0.75
e		0.50 BSC	
B	0.17	0.22	0.27
c	0.09	-	0.20
G	0.08	-	-
H		1.00 BSC	
J	0.08	-	0.20
K	0.20	-	-
2 H		2.00 BSC	

Ref. JEDEC - MO-136
ANAM AMKOR - DWG#34514





Ordering Information



INDEX

L

M

N

O

P

SECTION 4: Application Notes

CONTENTS

S5933 Device ID Assignment Procedure

Please use the following procedure to obtain S5933 Device IDs:

Send email to pci_id@amcc.com requesting an S5933 Device ID. Please include your:

- Name
- Company
- Address (with country)
- Phone number
- Email address
- Number of Device IDs requested. Prior approval of the PCI Applications Engineering Department is needed to obtain more than 10 Device IDs in any 6 month period.

- Your application (video adapter, sound adapter, etc.)
- Add-on microprocessor (if any)
- Operating systems used with your adapter (Windows® 95, Windows NT® 4.x , etc.)
- The PCI chipsets likely to be used with your adapter (Intel 440FX, Motorola MPC 106, etc.)
- The local rep or distributor you normally contact for AMCC technical information and data books
- Comments about AMCC's PCI products, requested features, etc.

You will normally receive your S5933 Device ID within 2 to 3 days via return email.

If you do not have email, please call (619) 450-9333 and ask the operator for the "PCI ID Desk."

If you have PCI technical questions, please email us at:
pci_techsupport@amcc.com
or visit our web site at http://www.amcc.com for an up-to-date list of AMCC email addresses.

We at AMCC appreciate your business!

AMCC PCI Applications Engineering Department

OVERVIEW

The AMCC S5933 PCI Controller provides add-in cards with bus mastering capabilities on the PCI bus. The S5933 internal FIFO is used to transfer data between the add-on and the PCI bus as a PCI initiator (bus master). The S5933 allows burst transfers at the full PCI bandwidth.

This application note discusses how the S5933 may be used in a PCI bus mastering (DMA) application. A brief background of DMA as it relates to the ISA bus is provided in Section 2. The remaining sections describe the bus master capabilities of the S5933 including add-on hardware support and required software support. Performance for bus mastering applications on the PCI bus is also discussed. An example C-Language program is provided in Appendix A to set up PCI DMA transfers with the S5933 controller.

DMA BACKGROUND

In ISA bus-based personal computers, there were two potential bus masters: the host CPU and the system DMA controller. There was no protocol defined for alternate bus masters to gain control of the bus and transfer data to other cards or platform memory. Most ISA add-in cards were designed as ISA bus slaves. The DMA controller or the CPU were used for data transfers to and from ISA cards.

For performance reasons, some ISA add-in cards required bus mastering capabilities. To become an ISA bus master, a free DMA channel was utilized. The ISA card asserted a DMA request to the 8237 DMA controller, the 82C37 asserted the HOLD input to the CPU. When the 82C37 received the HLDA acknowledge back from the CPU, it asserted an acknowledge to the add-in card. The add-in card then had control of the ISA bus to perform data transfers. For this reason, bus mastering is also referred to as DMA.

The PCI bus protocol has specific provisions to allow bus mastering (DMA) by any device connected to the PCI bus. If add-in cards perform their own data transfers, the host CPU is freed to perform other tasks (code execution, etc.). The PCI bus provides a dedicated request (REQ#)/grant(GNT#) pair to each PCI bus device (or card slot). These are all routed to a central bus arbiter that determines which device controls the PCI bus, based on a predefined priority scheme.

S5933 ARCHITECTURE

The S5933 performs DMA (bus master) transfers on the PCI bus through its FIFO interface. It has two, independent FIFOs. Each FIFO is 8-deep by 32-bits wide. One is used to transfer data from the PCI bus to the add-on, and the other is used to transfer data from the add-on to the PCI bus. The S5933 only performs DMA transfers to and from memory-mapped targets.

Each FIFO has an associated address and transfer count register. These registers may be defined by either the host CPU or add-on logic (configurable at reset). Each FIFO has a programmable priority and management scheme. Each FIFO also has the ability to generate interrupts when the transfer count expires or error conditions occur during a PCI bus transfer.

Configuring the S5933 for DMA Transfers

A DMA (bus master) transfer may be set up by either the host CPU, or add-on logic. This is defined for the S5933 at reset and cannot change during operation. Initiating a DMA transfer involves setting up the source/destination addresses and transfer counts as well as enabling the transfer. The FIFO relative priorities and FIFO management schemes are always programmed by the host CPU.

If an external, non-volatile boot memory is used with the S5933, the contents of offset 45h define FIFO operation. The following bits functions are defined:

Bit 7 Bus Master Register Access

0 Address and transfer count registers only accessible from the add-on interface

1 Address and transfer count registers only accessible from the PCI interface (default)

Bit 6 RDFIFO# or RD# Operation

0 Synchronous Mode — RDFIFO# and RD# functions as enables

1 Asynchronous Mode — RDFIFO# and RD# functions as clocks (default)

Bit 5 WRFIFO# or WR# Operation

0 Synchronous Mode — WRFIFO# and WR# functions as enables

1 Asynchronous Mode — WRFIFO# and WR# functions as clocks (default)

If no external non-volatile boot memory is used with the S5933, the default configuration for bus mastering is for transfers to be set up by the host CPU. Bits 6 and 5 define how the add-on FIFO interface operates. The default configuration for FIFO read and write strobes is to be asynchronous to the PCI clock (provided to the add-on by the S5933 BPCLK output). For more information on the FIFO add-on bus interface, refer to the S5933 PCI Controller Data Book.

PCI Initiated DMA Transfers

If no external non-volatile boot device is used, or if location 45h, bit 7 is 1, the address and transfer count registers are only accessible from the PCI bus interface. This requires that the PCI host write these register to initiate a DMA transfer. In this configuration, the S5933 can be programmed to generate a PCI interrupt (INTA#) when the DMA transfer count reaches zero or when an error occurs during a DMA transfer (target or master abort conditions).

PCI initiated DMA transfers must be enabled through the Bus Master Control/Status Register (MCSR). The register is at offset 3Ch in the S5933 PCI Operation Registers. S5933 Read DMA transfers and Write DMA transfers have separate control bits and can be individually enabled. The enable bits are not automatically cleared upon completion of a DMA transfer. DMA transfers remain enabled until these bits are cleared by the host CPU.

Add-on Initiated DMA Transfers

If an external, **serial** non-volatile boot device is used and location 45h, bit 7 is 0, the address and transfer count registers are only accessible from the add-on bus interface. This requires that add-on logic write these register to initiate a DMA transfer. In this configuration, the S5933 can be programmed to generate an add-on interrupt (IRQ#) when the DMA transfer count reaches zero or when an error occurs during a DMA transfer. A serial boot device is required because the transfer enable inputs used for add-on initiated DMA are multiplexed with the byte-wide nv-memory interface.

Add-on initiated DMA transfers are enabled using the AMREN and AMWEN inputs. The bus master enable bits in the Bus Master Control/Status Register (MCSR) are ignored. Asserting AMREN enables the S5933 to request control of the PCI bus for a PCI read transfer when the appropriate FIFO conditions are met. Asserting AMWEN enables the S5933 to request control of the PCI bus for a PCI write transfer when the appropriate FIFO conditions are met (see section 3.4 on FIFO management schemes). If an enable is deasserted during a DMA transfer, the current PCI bus transaction completes and the S5933 deasserts REQ#, giving up control of the PCI bus.

DMA Address Registers

There are two DMA address registers: the Master Write Address Register (MWAR) and the Master Read Address Register (MRAR). These registers are located at offsets 24h and 2Ch, respectively, in the S5933 PCI Operation Registers. These are written with the beginning memory address of the DMA transfer. The S5933 requires that DMA transfers start on double-word boundaries (A1 and A0 = 0).

During a DMA transfer, the address registers are incremented by four after each completed data phase. If a PCI target disconnects and requests a retry from the S5933, the correct address is maintained to allow the transfer to begin from where it was disconnected.

DMA Transfer Count Registers

There are two DMA transfer count registers: the Master Write Transfer Count Register (MWTC) and the Master Read Transfer Count Register (MRTC). These registers are located at offsets 28h and 30h, respectively, in the S5933 PCI Operation Registers. These are written with the a DMA transfer byte count of up to 64 Mbytes. The transfer count registers are decremented by four after each completed data phase. The transfer count registers do not reset to their initial value after reaching zero, the PCI host must reprogram them.

Although S5933 DMA transfer must begin on double-word boundaries, the transfer count does not have to be a multiple of four bytes. When the transfer count decrements below four, only the byte enable corresponding to the number of bytes left are asserted. For example, if a transfer count of 10 was programmed into one of the transfer count registers, two data transfers would complete with all byte enables asserted (8 bytes). The final data transfer would have only BE0# and BE1# asserted, transferring the final two bytes.

For add-on initiated DMA transfers, the transfer counts are enabled or disabled through the Add-on General Control/Status Register (AGCSTS), bit 28. The transfer counts for read and write transfers cannot be individually enabled. This may be useful in applications where the amount of data to be transferred is not known. If transfer counts are disabled, the S5933 continues to transfer data according to the conditions listed above, but transfer counts are ignored.

FIFO Management Schemes

The S5933 provides flexibility in how the FIFO is managed for DMA transfers. The FIFO management scheme determines when the S5933 requests the PCI bus (asserts REQ#). The most efficient way to untilize the capabilities of the PCI bus is with burst transfers. Requesting the PCI bus every time the FIFO contains a single double-word is an ineffecient use of the bus, and limits the performance of other PCI devices within a system. It is more desirable request the bus when multiple operations are required, allowing the S5933 to perform a burst transfer.

The management scheme is configurable for the PCI to add-on and add-on to PCI FIFOs (and may be different for each). Bus mastering must be enabled for the management scheme to apply (via the MCSR enable bits or AMREN/AMWEN). The FIFO management option is programmed through the Bus Master Control/Status Register (MCSR).

For the PCI to add-on FIFO (DMA reads), there are two options. The FIFO can be programmed to request the bus when any FIFO location is empty or only when four or more locations are empty. After the S5933 is granted control of the PCI bus, the management scheme does not apply. The device continues to read as long as there is an open FIFO location. For DMA read transfers, the S5933 maintains control of the PCI bus until one of the following events:

- The read transfer count (MRTC) reaches zero
- Bus mastering is disabled (with the MSCR enable bit or AMREN)
- Another master requests the bus and the Latency Timer is expired
- The PCI target aborts the transfer
- The PCI to add-on FIFO becomes full

For the add-on to PCI FIFO (DMA writes), there are two management options. The FIFO can be programmed to request the bus when any FIFO location is full or only when four or more locations are full. After the S5933 is granted control of the PCI bus, the management scheme does not apply. The device continues to write as long as there is data in the FIFO. For DMA write transfers, the S5933 maintains control of the PCI bus until one of the following events:

- The write transfer count (MWTC) reaches zero
- Bus mastering is disabled (with the MSCR enable bit or AMWEN)
- Another master requests the bus and the Latency Timer is expired
- The PCI target aborts the transfer
- The add-on to PCI FIFO becomes empty

There are two special cases for the add-on to PCI FIFO management scheme. The first case is when the FIFO is programmed to request the PCI bus only when four or more locations (16 bytes) are full, but the transfer count is less than 16 bytes. In this situation, the FIFO ignores the management scheme and finishes transferring the data. The second case is when the S5933 is configured for add-on initiated bus mastering. In this situation, the FIFO management scheme must be set to request the PCI bus when one or more locations are full.

S5933 DMA Channel Priority

In many applications, the S5933 performs both DMA read and write transfers. This requires a priority scheme be implemented between the two FIFOs. If the FIFO management condition for initiating a PCI read and a PCI write are both met, a method must exist to determine which transfer is performed first.

Bits D12 and D8 in the Bus Master Control/Status Register (MCSR) control the read and write DMA channel priority, respectively. If these bits are both set or both clear, priority alternates, beginning with read transfers. If the read priority is set and the write priority is clear, read cycles take priority. If the write priority is set and the read priority is clear, write cycles take priority. Priority arbitration is only done when neither FIFO has control of the PCI bus (the PCI to add-on FIFO never interrupts an add-on to PCI FIFO transfer in progress and vice-versa).

S5933 DMA Interrupts

The S5933 can generate interrupts under the following conditions: the read transfer count reaches zero, the write transfer count reaches zero, or an error occurs on the PCI bus during a DMA transfer.

Which interface (PCI bus or add-on bus) receives the interrupt is determined by which side initiated the DMA transfer. If PCI initiated DMA transfers are used, a PCI INTA# interrupt is generated when an interrupt condition is met. If add-on initiated DMA transfers are used, IRQ# is generated to the add-on interface. Interrupts are optional and may be disabled.

Transfer Count Interrupts

Transfer count interrupts may come from two sources: read transfer count and/or write transfer count. One or both interrupts may be enabled, or both may be disabled. A read transfer count interrupt is generated when the Master Read Transfer Count Register (MRTC) decrements to zero. A write transfer count interrupt is generated when the Master Write Transfer Count Register (MWTC) decrements to zero. These registers decrement when a PCI transfer completes successfully.

If add-on initiated DMA transfers are used with transfer counts disabled, these interrupt sources are disabled.

PCI Bus Error Condition Interrupts

In some situations, the PCI bus may signal an error condition during an S5933 DMA transfer. These error conditions include a target abort or master abort on the PCI bus. If one of these conditions exists, it is important to notify the device which initiated the transfer that it cannot complete successfully. Interrupts on PCI error conditions are only enabled if one or both of the transfer count interrupts are enabled. There is no individual enable bit for PCI error interrupts.

A PCI target abort indicates an error condition where no number of retries to the target will result in a successful data transfer. A PCI master abort occurs when a PCI bus master (the S5933, in this case) attempts to access a PCI target which is either nonexistent or disabled. In either of these situations, the device which set up the DMA transfer is notified with an interrupt (either INTA# or IRQ#).

Controlling S5933 DMA Interrupts

For PCI initiated DMA transfers, interrupts are enabled through the S5933 Interrupt Control Status Register (INTCSR). This register is located at offset 38h in the S5933 PCI Operation Registers. INTCSR is also accessed during interrupt service routines to determine the interrupt source and clear the interrupt. The following INTCSR bits relate to DMA Operations:

Bit 14 **Enable Interrupt on Write Transfer Complete.** If set, INTA# is generated when MWTC decrements to zero during a DMA transfer.

Bit 15 **Enable Interrupt on Read Transfer Complete**. If set, INTA# is generated when MRTC decrements to zero during a DMA transfer.

Bit 18 **Write Transfer Complete Interrupt**. When set, this bit indicates MWTC has decremented to zero and INTA# has been asserted. Writing a one to this bit clears the interrupt source and deasserts INTA#. Writing a zero to this bit has no effect.

Bit 19 **Read Transfer Complete Interrupt**. When set, this bit indicates MRTC has decremented to zero and INTA# has been asserted. Writing a one to this bit clears the interrupt source and deasserts INTA#. Writing a zero to this bit has no effect.

Bit 20 **Master Abort Interrupt**. When set, this bit indicates that the S5933 had to perform a master abort and INTA# has been asserted. Writing a one to this bit clears the interrupt source and deasserts INTA#. Writing a zero to this bit has no effect.

Bit 21 **Target Abort Interrupt**. When set, this bit indicates that the S5933 received a target abort and INTA# has been asserted. Writing a one to this bit clears the interrupt source and deasserts INTA#. Writing a zero to this bit has no effect.

For add-on initiated DMA transfers, interrupts are enabled through the S5933 Add-on Interrupt Control/Status Register (AINT). This register is located at offset 38h in the S5933 Add-on Operation Registers. AINT is also accessed during interrupt service routines to determine the interrupt source and clear the interrupt. The following AINT bits relate to DMA Operations:

Bit 14 **Enable Interrupt on Write Transfer Complete**. If set, IRQ# is generated when MWTC decrements to zero during a DMA transfer.

Bit 15 **Enable Interrupt on Read Transfer Complete**. If set, IRQ# is generated when MRTC decrements to zero during a DMA transfer.

Bit 18 **Write Transfer Complete Interrupt**. When set, this bit indicates MWTC has decremented to zero and IRQ# has been asserted. Writing a one to this bit clears the interrupt source and deasserts IRQ#. Writing a zero to this bit has no effect.

Bit 19 **Read Transfer Complete Interrupt**. When set, this bit indicates MRTC has decremented to zero and IRQ# has been asserted. Writing a one to this bit clears the interrupt source and deasserts IRQ#. Writing a zero to this bit has no effect.

Bit 21 **Bus Master Error Interrupt**. When set, this bit indicates that the S5933 had to perform a master abort or received a target abort and IRQ# has been asserted. Writing a one to this bit clears the interrupt source and deasserts IRQ#. Writing a zero to this bit has no effect.

PCI Interrupt Considerations

If the S5933 is configured to generate PCI bus interrupts (INTA#) for DMA transfer counts and PCI bus error conditions, considerations must be made for the interrupt handler. The interrupt vector must be obtained, and because hardware interrupts may be shared, interrupt handlers may have to be "chained."

Enabling PCI Interrupts

The Interrupt Pin (INTFIN) PCI Configuration Register may be loaded out of non-volatile memory offset 7Dh by the S5933 at reset. The value loaded into this register identifies which PCI interrupt: INTA#, INTB#, INTC#, or INTD# is used. The default value is 01h, identifying INTA#. For the S5933, the INTA# output should be connected to the PCI bus INTA# pin. If PCI interrupts are not required, this register may be initialized to 00h (interrupts disabled) or ignored.

The four PCI interrupts are mapped within the system chipset to standard PC IRQ numbers (0-15). The 82C59A compatible interrupt controllers (two cascaded controllers) each have an interrupt mask register. The master mask register, located at system I/O location 21h controls the masking of IRQ0-7, and the slave mask register, located at system I/O location 1Ah controls the masking of IRQ8-15. Application software must make sure the S5933 interrupt line (indicated by the PCI Interrupt Line Register is unmasked. If the interrupt is masked, the handler never executes.

PCI Host Interrupt Handlers

The S5933 Interrupt Line (INTLN) PCI Configuration Register is loaded out of non-volatile memory offset 7Ch by the S5933 at reset. The value loaded into the register is a hardware interrupt number for the host interrupt controller (IRQ0 to 15). This value may be used by the host, or the BIOS may overwrite it with its own value.

Multiple PCI devices may be assigned to a single hardware interrupt by the host. PCI device drivers are, therefore, required to determine if the current interrupt was generated by the device it services. If not, it must pass control, or "chain" the previous interrupt handler to service the interrupt.

Each application's code must install its own interrupt vector. To do this, the Interrupt Line Configuration Register is read. A value between 0 and 15 is returned, corresponding to a PC hardware IRQ number. This must be translated into a software interrupt number. The following table shows the conversions for a PC:

Hardware Interrupt	Software Interrupt
IRQ0	08h
IRQ1	09h
IRQ2	0Ah
IRQ3	0Bh
IRQ4	0Ch
IRQ5	0Dh
IRQ6	0Eh
IRQ7	0Fh
IRQ8	70h
IRQ9	71h
IRQ10	72h
IRQ11	73h
IRQ12	74h
IRQ13	75h
IRQ14	76h
IRQ15	77h

Once the software interrupt number is calculated, the previous interrupt handler's vector can be read and stored. The new interrupt vector is then installed. In this manner, numerous devices within a system can share a single hardware interrupt.

Each application's interrupt handler must first check the source of the interrupt. For S5933 applications, this is done by reading the S5933 Interrupt Control/ Status Register. The status of all possible S5933 PCI interrupt sources is indicated by this register. If the status bits indicate that no interrupts were generated by the S5933, the handler must call the previous interrupt handler whose vector it replaced. The previous interrupt handler then performs a similar task for the device it services, and this process continues until the device which generated the interrupt is found.

If the interrupt handler determines that the source of the interrupt was the S5933, the interrupt source must be cleared through the Interrupt Control/Status Register (INTCSR). The handler then performs whatever tasks are necessary to service the interrupt (such as rewriting address or transfer count registers). Finally, the handler must clear the 83C59A interrupt controller 'in-service' bit. This bit is set when the host processor acknowledges the interrupt and jumps to the interrupt service routine. A specific end-of-interrupt (EOI) is used to clear this bit. If a PCI device is mapped to hardware interrupts IRQ8 to IRQ15, two EOI commands must be issued. One EOI must be issued for the slave 82C59A which supports IRQ8-15. A second EOI is required for the master 82C59A. The second EOI is a specific EOI for IRQ2 because the slave interrupt controller in a PC is cascaded into the IRQ2 input of the master interrupt controller. Without the end-of-interrupt sequence, the interrupt controllers will not recognize further interrupts from that source.

PCI DMA PERFORMANCE FACTORS

There are a number of factors which determine DMA performance on the PCI bus. The clock speed and data bus width are important in determining maximum bus bandwidth. The most important factor in DMA performance is traffic on the PCI bus. As the number of PCI devices which require access to the bus increases, the bandwidth available to each individual device decreases.

PCI Bus Arbitration

Each device on the PCI bus has a dedicated REQ#/ GNT# pair which are connected to the system bus arbiter. When a device asserts REQ#, it indicates a data transfer is required. The transfer may be a single data phase or a burst. The bus arbiter asserts GNT# to the device to indicate that it may now perform the transfer.

The bus arbiter may remove GNT# from a PCI bus master on any PCI clock. The current transaction completes, and the PCI master gives up control of the bus. GNT# may already be asserted to the next master, but it is not allowed to drive the PCI bus until IRDY# and FRAME# are deasserted, indicating the bus is idle. If the original bus master has more data to be transferred, it may keep REQ# asserted, but must wait for GNT# again.

The priority scheme used to determine which PCI device controls the bus at a given time is determined by the system. The PCI specification requires a few extra Configuration Registers within PCI bus master devices. During system initialization, these registers are read to determine each device's requirements (if any). Based on these, a priority scheme can be defined which is unique to that system. Ideally, the arbitration scheme gives priority to devices with higher bandwidth requirements, but does not prevent other PCI bus masters from gaining control of the bus.

PCI Bus Access Latency

There are three components to latency on the PCI bus. The total latency is measured from the time a bus master requests the bus (asserts REQ#) to when the target of the transfer completes the transfer (asserts TRDY#). Each of these components is described in the following sections.

Arbitration Latency

Once a PCI device asserts REQ#, it must wait for the bus arbiter to assert GNT#. This delay is called arbitration latency. This is determined by the priority scheme used by the bus arbiter, the relative priority of the device requesting the bus, and the amount of activity on the PCI bus.

Bus Acquisition Latency

Once a PCI device receives GNT# from the arbiter, it must wait for the current bus master to complete it's transaction (indicated by FRAME# and IRDY# deasserted) before driving the PCI bus. Bus acquisition latency is the time from when GNT# is received to when FRAME# is asserted by a particular master. This is determined by the length of the current bus master's transfer.

Target Latency

After a PCI device begins a bus cycle, it must wait for the target of the transfer to complete the cycle (by asserting TRDY#). Target latency is the time from when FRAME# is asserted to when TRDY# is asserted. This is unique for each target and depends

on the function of the target (main memory, VGA controller, network interface, etc.).

PCI Configuration Registers

A PCI device with bus mastering capabilities may implement up to three additional configuration registers. These registers indicate the requirements of a particular PCI device. These are read during system initialization and may be used to define a bus master priority scheme. The S5933 implements all of these registers. They can be boot loaded from an external, non-volatile memory device at reset.

Latency Timer Register

If a bus master is capable of PCI bursts of more than two data phases, this register is required. The Latency Timer Register defines the number of PCI clocks that the S5933 is guaranteed control of the bus. The Latency Timer Register is shown in Figure 1. The value programmed in this register is decremented once every 8 PCI clocks after the S5933 asserts FRAME#.

The default value for this register is 00h, indicating that the S5933 has no minimum transfer length requirement. The system can overwrite the Latency Timer Register with any value. This prevents an individual PCI master from controlling the bus for an extremely long period.

If no other PCI master has requested the bus, the Latency Timer value does not matter (even if it has expired). The S5933 transfers data until the transaction is complete. If another bus master requests the bus and receives GNT# while the S5933 is transferring data, three situations can occur:

1) If the Latency Timer has not expired, and the S5933 completes its transaction before it expires, the transaction completes normally.
2) If the Latency Timer has not expired, the S5933 transfers data until the latency timer expires where it completes the current data phase, terminates the transaction and gives up control of the bus
3) If the Latency Timer has already expired, the S5933 completes the current data phase, terminates the transaction, and gives up control of the bus.

Minimum Grant Register

This register defines how long of a burst period the device typically requires (in units of 250 ns). This read-only register is for information only. It may be used by the system, in conjunction with the Maximum Latency register, to define a PCI bus arbitration scheme (if the bus arbiter is programmable).

A value of zero (default for the S5933) indicates there is no strict requirement for burst length. The Minimum Grant Register is shown in Figure 2.

Figure 1. Latency Timer Register Definition

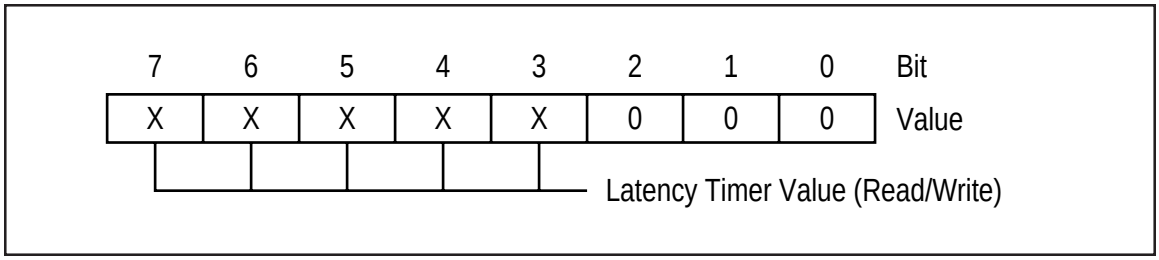


Figure 2. Minimum Grant Register Definition

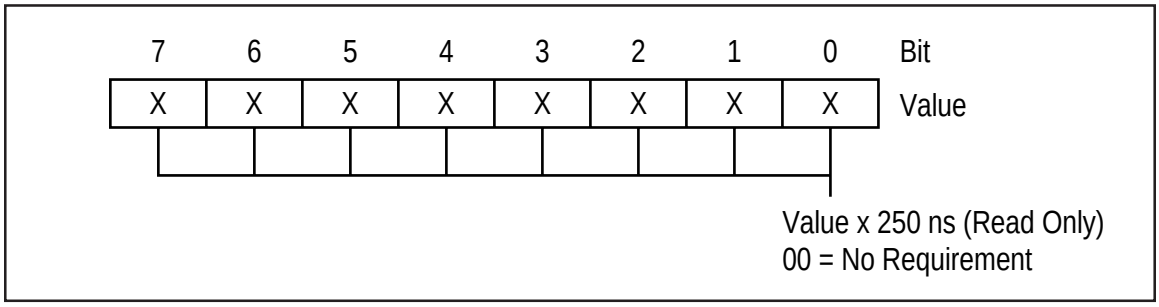
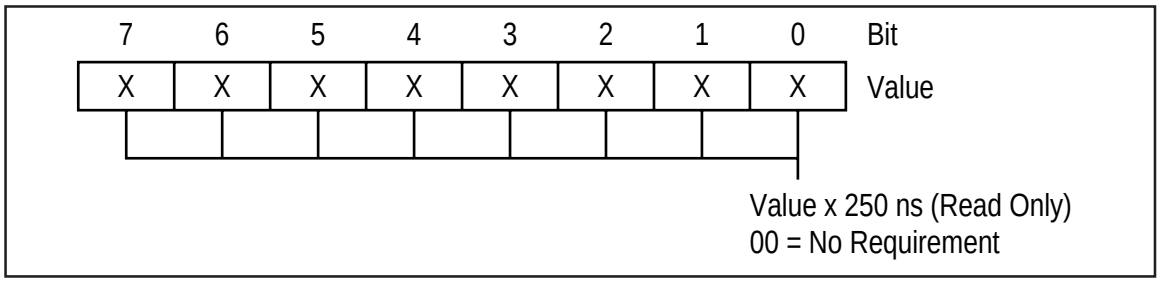


Figure 3. Maximum Latency Register Definition



Maximum Latency Register

This register defines how often the device typically needs PCI bus access (in units of 250 ns). This read-only register is for information only. It may be used by the system, in conjunction with the Minimum Grant register, to define a PCI bus arbitration scheme (if the bus arbiter is programmable).

A value of zero (default for the S5933) indicates there is no strict requirement for access to the PCI bus. The Maximum Latency Register is shown in Figure 3.

Sample PCI Performance Calculation

The maximum theoretical bandwidth for the PCI bus at 33 MHz is 132 Mbytes per second. The actual bandwidth is less. Achievable bandwidth depends of factors such as bus utilization by other masters, the bus arbitration scheme, and burst length limitations of both the PCI initiator and target. The following examples show bandwidth calculations for two situations: a DMA transfer from the S5933 to main DRAM memory, and a DMA transfer to another S5933 add-in board. These performance calculations are based on the current PCI systems. As chipsets, memory controllers and other PCI devices evolve, performance will increase, accordingly.

S5933 Burst to Main DRAM Memory

Table 1 shows a situation where an add-on device can write data to the S5933 FIFO at a rate of one double-word (32-bits) every 60 ns. The target for the DMA transfer is main DRAM memory. The PCI memory controller allows 4 data phase write bursts. The fifth data phase receives a target requested retry. The PCI Specification requires that a PCI initiator deassert REQ# for two clocks. For this example, a 4 clock latency period is used from the reassertion of REQ# by the S5933 to the reassertion of GNT# by the PCI bus arbiter.

The sequence shown assumes the following initial conditions:

- Master Write Address Register (MWAR) = 100000h
- Master Write Transfer Count Register (MWTC) is disabled
- Bus mastering for the S5933 is already enabled
- The Add-on to PCI FIFO is full (8 dwords)
- The PCI bus arbiter has just asserted GNT# to the S5933

Table 1. Sample S5933 Burst to Main Memory

Time	PCI Bus Activity	Add-on Bus Activity	FIFO Status	REQ#	GNT#
30 ns	Address = 100000h	Idle	8 dwords	0	0
60 ns	Data Transfer 1	Wait State	7 dwords	0	0
90 ns	Data Transter 2	Write FIFO	7 dwords	0	0
120 ns	Data Transfer 3	Wait State	6 dwords	0	0
150 ns	Data Transfer 4	Write FIFO	6 dwords	0	0
180 ns	Target Disconnect	Wait State	6 dwords	0	0
210 ns	Idle (or other master)	Write FIFO	7 dwords	1	1
240 ns	Idle (or other master)	Wait State	7 dwords	1	1
270 ns	Idle (or other master)	Write FIFO	8 dwords	0	1
300 ns	Idle (or other master)	Idle	8 dwords	0	1
330 ns	Idle (or other master)	Idle	8 dwords	0	1
360 ns	Idle (or other master)	Idle	8 dwords	0	1
390 ns	Idle (or other master)	Idle	8 dwords	0	0
420 ns	Address = 100010h	Idle	8 dwords	0	0
450 ns	Data Transfer 5	Wait State	7 dwords	0	0

The sequence would repeat every 390 ns with a 33 MHz PCI bus clock. Four Dwords, or 16 bytes have been transferred to main memory in during this time. This would average out to about 40 MBytes/sec.

The major factor in the calculation is how long it takes for the S5933 to regain control of the PCI bus after the main memory controller disconnects after the fourth data phase. This depends on the number of PCI devices using the bus. Because the transfer is to main memory, the S5933 must compete for bus bandwidth with numerous other devices that are usually on the Primary PCI bus. A typical system may have the host CPU, a VGA controller, an ethernet interface, and a SCSI or IDE drive sharing the Primary PCI bus. If a rotational priority scheme were implemented by the bus arbiter, it could be significantly more than 4 PCI clocks before the S5933 regains control of the bus.

S5933 Burst to Another S5933 PCI Card

Table 2 shows a situation where an add-on device can write data to the S5933 FIFO at a rate of one double-word (32-bits) every 60 ns. The target for the DMA transfer is the pass-thru interface of another S5933 device which can accept data at a rate of one double-word every 30 ns. The S5933 pass-thru interface does not disconnect from a burst write unless the add-on has not read the pass-thru data register within 16 PCI clocks for the first data phase or 8 PCI clocks on successive data phases. In this situation, the initiator S5933 deasserts request because it runs out of data. The target S5933 never disconnects in this situation.

The sequence shown assumes the following initial conditions:

- Master Write Address Register (MWAR) = 100000h
- Master Write Transfer Count Register (MWTC) is disabled
- Bus mastering for the S5933 is already enabled
- The Add-on to PCI FIFO is full (8 dwords)
- The PCI bus arbiter has just asserted GNT# to the S5933

In this example, it is assumed that the AMWEN signal has been deasserted when the FIFO goes empty at 480 ns. This allows the FIFO to refill before another transfer is initiated. This is the most efficient way to utilize the PCI bus. Sending a signal, long burst is better than sending numerous short bursts because less overall time is spent arbitrating for PCI bus control.

The FIFO would be full again at 690 ns. If AMWEN is reasserted when the FIFO is full, and a 4 clock latency is assumed (as with the previous example), the next address phase begins at 810 ns. The process would repeat from there. This results in 15 dwords (60 bytes) being transferred every 810 ns. The results in an average 74 MByte/sec. transfer rate. Again, this is heavily dependent on PCI bus utilization by other devices. Unless the two cards in question share an isolated PCI bus on the secondary side of a PCI to PCI bridge, bandwidth would likely be less than this.

Table 2. Sample S5933 Burst to Another S5933 Device

Time	PCI Bus Activity	Add-on Bus Activity	FIFO Status	REQ#	GNT#
30 ns	Address = 100000h	Idle	8 dwords	0	0
60 ns	Data Transfer 1	Wait State	7 dwords	0	0
90 ns	Data Transter 2	Write FIFO	7 dwords	0	0
120 ns	Data Transfer 3	Wait State	6 dwords	0	0
150 ns	Data Transfer 4	Write FIFO	6 dwords	0	0
180 ns	Data Transfer 5	Wait State	5 dwords	0	0
210 ns	Data Transter 6	Write FIFO	5 dwords	0	0
240 ns	Data Transfer 7	Wait State	4 dwords	0	0
270 ns	Data Transfer 8	Write FIFO	4 dwords	0	0
300 ns	Data Transfer 9	Wait State	3 dwords	0	0
330 ns	Data Transter 10	Write FIFO	3 dwords	0	0
360 ns	Data Transfer 11	Wait State	2 dwords	0	0
390 ns	Data Transfer 12	Write FIFO	2 dwords	0	0
420 ns	Data Transfer 13	Wait State	1 dword	0	0
450 ns	Data Transter 14	Write FIFO	1 dword	0	0
480 ns	Data Transfer 15	Wait State	0 dwords	1	1
510 ns	Idle (or other Master)	Write FIFO	1 dword	1	1

DMA SOFTWARE SUPPORT

DMA transfers with the S5933 depend mostly on hardware. Most of the design is the interface on the add-on card which fills and empties the FIFO. There is some software support required for DMA transfers. Address and transfer count registers must be loaded, the FIFOs must be configured, and interrupts (if used) must be enabled and serviced. The following sections provide an overview of what actions are required by software for S5933 DMA operations.

PCI Initiated DMA Transfers

For PCI initiated DMA transfers, the PCI host CPU (Pentium™, Alpha™, etc.) sets up the S5933 to perform bus master transfers. The following tasks must be completed to setup FIFO bus mastering:

1) **Define interrupt capabilities**. The PCI to add-on and/or add-on to PCI FIFO can generate a PCI interrupt to the host when the transfer count reaches zero.

INTCSR	Bit 15	Enable Interrupt on read transfer count equal zero
INTCSR	Bit 14	Enable Interrupt on write transfer count equal zero

2) **Reset FIFO flags**. This may not be necessary, but if the state of the FIFO flags is not known, they should be initialized.

MCSR	Bit 26	Reset add-on to PCI FIFO flags
MCSR	Bit 25	Reset PCI to add-on FIFO flags

3) **Define FIFO management scheme**. These bits define what FIFO condition must exist for the PCI bus request (REQ#) to be asserted by the S5933.

MCSR	Bit 13	PCI to add-on FIFO management scheme
MCSR	Bit 9	Add-on to PCI FIFO management scheme

4) **Define FIFO priority scheme**. These bits determine which FIFO has priority if both meet the defined condition to request the PCI bus. If these bits are the same, priority alternates, with read accesses occurring first.

MCSR	Bit 12	Read vs. write priority
MCSR	Bit 8	Write vs. read priority

5) **Define transfer source/destination address**. These registers are written with the first address that is to be accessed by the S5933. These address registers are updated after each access to indicate the next address to be accessed. Transfers must start on DWORD boundaries.

MWAR	All	Bus master write address
MRAR	All	Bus master read address

6) **Define transfer byte counts**. These registers are written with the number of bytes to be transferred. The transfer count does not have to be a multiple of four bytes. These registers are updated after each transfer to reflect the number of bytes remaining to be transferred.

MWTC	All	Write transfer byte count
MRTC	All	Read transfer byte count

7) **Enable Bus Mastering**. Once steps 1-6 are completed, the FIFO may operate as a PCI bus master. Read and write bus master operations may be independently enabled or disabled.

MCSR	Bit 14	Enable PCI to add-on FIFO bus mastering
MCSR	Bit 10	Enable add-on to PCI FIFO bus mastering

It is recommended that bus mastering be enabled as the last step. Some applications may choose to leave bus mastering enabled and start transfers by writing a non-zero value to the transfer count registers. This also works, provided the entire 26-bit transfer count is written at once. If transfer count interrupts are enabled, they must be enabled after the transfer count(s) are written. If interrupts are enabled and the transfer count is zero, an interrupt occurs immediately.

Servicing a PCI Initiated DMA Transfer Interrupt

If interrupts are enabled, a host interrupt service routine is also required. The service routine determines the source of the interrupt and resets the interrupt. The source of the interrupt is indicated in the PCI Interrupt Control/Status Register (INTCSR). Typically, the interrupt service routine is used to set up the next transfer by writing a new address and transfer count value, but some applications may also require other actions. If read transfer or write transfer complete interrupts are enabled, master and target abort interrupts are automatically enabled. Writing a one to these bits clears the corresponding interrupt.

INTCSR	Bit 21	Target abort caused interrupt
INTCSR	Bit 20	Master abort caused interrupt
INTCSR	Bit 19	Read transfer complete caused interrupt
INTCSR	Bit 18	Write transfer complete caused interrupt

Add-on Initiated DMA Transfers

For add-on initiated DMA transfers, the add-on sets up the S5933 to perform bus master transfers. The following tasks must be completed to setup FIFO bus mastering:

1) **Define transfer count abilities**. For add-on initiated bus mastering, transfer counts may be either enabled or disabled. Transfer counts for read and write operations cannot be individually enabled.

AGCSTS	Bit 28	Enable transfer count for read and write bus master transfers

2) **Define interrupt capabilities**. The PCI to add-on and/or add-on to PCI FIFO can generate an interrupt to the add-on when the transfer count reaches zero (if transfer counts are enabled).

AINT	Bit 15	Enable interrupt on read transfer count equal zero
AINT	Bit 14	Enable interrupt on write transfer count equal zero

3) **Reset FIFO flags**. This may not be necessary, but if the state of the FIFO flags is not known, they should be initialized.

AGCSTS	Bit 26	Reset add-on to PCI FIFO flags
AGCSTS	Bit 25	Reset PCI to add-on FIFO flags

4) **Define FIFO management scheme**. These bits define what FIFO condition must exist for the PCI bus request (REQ#) to be asserted by the S5933. This must be programmed through the PCI interface.

MCSR	Bit 13	PCI to add-on FIFO management scheme
MCSR	Bit 9	Add-on to PCI FIFO management scheme

5) **Define FIFO priority scheme**. These bits determine which FIFO has priority if both meet the defined condition to request the PCI bus. If these bits are the same, priority alternates, with read accesses occurring first. This must be programmed through the PCI interface.

MCSR	Bit 12	Read vs. write priority
MCSR	Bit 8	Write vs. read priority

6) **Define transfer source/destination address**. These registers are written with the first address that is to be accessed by the S5933. These address registers are updated after each access to indicate the next address to be accessed. Transfers must start on DWORD boundaries.

MWAR	All	Bus master write address
MRAR	All	Bus master read address

7) **Define transfer byte counts**. These registers are written with the number of bytes to be transferred. The transfer count does not have to be a multiple of four bytes. These registers are updated after each transfer to reflect the number of bytes remaining to be transferred. If transfer counts are disabled, these registers do not need to be programmed.

MWTC	All	Write transfer byte count
MRTC	All	Read transfer byte count

8) **Enable Bus Mastering**. Once steps 1-7 are completed, the FIFO may operate as a PCI bus master. Read and write bus master operation may be independently enabled or disabled. The AMREN and AMWEN inputs control bus master enabling for add-on initiated bus mastering. The MCSR bus master enable bits are ignored for add-on initiated bus mastering.

It is recommended that bus mastering be enabled as the last step. Some applications may choose to leave bus mastering enabled (AMREN and AMWEN asserted) and start transfers by writing a non-zero value to the transfer count registers (if they are enabled). This works, provided the entire 26-bit transfer count is written at once. If transfer count interrupts are enabled, they must be enabled after the transfer count(s) are written. If interrupts are enabled and the transfer count is zero, an interrupt occurs immediately.

Servicing an Add-on Initiated DMA Transfer Interrupt

If interrupts are enabled, an add-on CPU interrupt service routine is also required. The service routine determines the source of the interrupt and resets the interrupt. The source of the interrupt is indicated in the Add-on Interrupt Control Register (AINT). Typically, the interrupt service routine is used to set up the next transfer by writing a new address and transfer count value (if enabled), but some applications may also require other actions. If read transfer or write transfer complete interrupts are enabled, the master/target abort interrupt is automatically enabled. Writing a one to these clears the corresponding interrupt.

AINT	Bit 21	Master/target abort caused interrupt
AINT	Bit 19	Read transfer complete caused interrupt
AINT	Bit 18	Write transfer complete caused interrupt

SAMPLE S5933 DMA SUPPORT CODE

The following section is a sample program written in C-language to setup the S5933 for DMA operations. The code is written for PCI initiated bus mastering using transfer count interrupts. The code is written for an x86 compatible PCI platform, but could be modified to support other host processors.

The code has been compiled using Borland C/C++ Version 4.5. Because 32-bit registers are used within the code, code must be compiled to generate 386 code (otherwise, 32-bit register mneumonics such as _EAX are not recognized).

```
#include <dos.h>
#include <stddef.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include “amcc.h”

/*****************************************************************/
/*  A2P = Add-on to PCI FIFO                                     */
/*  P2A = PCI to Add-on FIFO                                     */
/*****************************************************************/

/* Transfer Count Interrupt Enables */
#define EN_READ_TC_INT        0x00008000L
#define EN_WRITE_TC_INT       0x00004000L

/* FIFO Flag Reset */
#define RESET_A2P_FLAGS       0x04000000L
#define RESET_P2A_FLAGS       0x02000000L

/* FIFO Management Scheme */
#define A2P_REQ_AT_4FULL      0x00000200L
#define P2A_REQ_AT_4EMPTY     0x00002000L

/* FIFO Relative Priority */
#define A2P_HI_PRIORITY       0x00000100L
#define P2A_HI_PRIORITY       0x00001000L

/* Enable Transfer Count */
#define EN_TCOUNT             0x10000000L

/* Enable Bus Mastering */
#define EN_A2P_TRANSFERS      0x00000400L
#define EN_P2A_TRANSFERS      0x00004000L

/* Identify S5933 Interrupt Sources */
#define ANY_S5933_INT         0x80
#define READ_TC_INT           0x08
#define WRITE_TC_INT          0x04
#define MASTER_ABORT_INT      0x10
#define TARGET_ABORT_INT      0x20
#define BUS_MASTER_INT        0x20

/* Global Variable Definition */
byte    interrupt_line;
word    op_reg_base_address;
void(interrupt *oldhandler)(void);
```

```
/*****************************************************************/
/*      MAIN Code Segment                                        */
/*****************************************************************/

void main()
{
        void    setup_pci_dma(void);
        void    setup_int_vect(byte bus_num,byte dev_func);
        word    vendor_id, device_id, index;

        byte    bus_num, dev_func;
        int     bios_present;


/* AMCC Default Vendor/Device ID's */
        vendor_id = 0x10E8;
        device_id = 0x4750;
        index = 0;

/* Disable Host Interrupts */
        disable();

/* Look for a valid PCI BIOS */
        if(pci_bios_present(NULL,NULL,NULL)==SUCCESSFUL){
                bios_present=TRUE;
                printf("PCI BIOS Found\n\n");
        }
        else{
                printf("PCI BIOS not present\n\n");
                bios_present=FALSE;
        }

/* If the BIOS is present, look for the AMCC device */
        if(bios_present){
                if(find_pci_device(device_id,vendor_id,index,&bus_num,
                        &dev_func)==SUCCESSFUL){
                        printf("AMCC Device Found: Bus=%d Device=%d Function=%d\n\n",
                                bus_num,(dev_func>>3),(dev_func&0x7));
                }
                else{
                        printf("AMCC Device Not Found\n\n");
                }
        }

/* Find the physical location of the S5933 in I/O space */
        read_configuration_word(bus_num,dev_func,
                 PCI_CS_BASE_ADDRESS_0,&op_reg_base_address);

/* Mask Lower 2 bits of BADR0 */
        op_reg_base_address = (op_reg_base_address & 0xFFFC);

/* Call routine to install interrupt vector */
        setup_int_vect(bus_num,dev_func);

/*Enable hardware interrupts */
        enable();

/* Call routine to setup PCI initiated bus mastering */
        setup_pci_dma();
}
```

```
/*************************************************************************/
/*      Function:      setup_pci_dma                                     */
/*      Purpose:       Configure the S5933 for PCI initiated DMA         */
/*      Inputs:        None                                              */
/*      Outputs:       None                                              */
/*                                                                       */
/* The following function assumes the S5933 is configured                */
/* for PCI initiated DMA transfers and sets up the DMA channels          */
/*************************************************************************/

void setup_pci_dma(void)
{
        char src[20], dest[20], rtc[20], wtc[20];
        dword  source, destination, temp = 0;
        dword readtc, writetc;

        /* Read in source,destination and transfer counts */
        printf("Input address values in hex format: 0x...\n\n");
        printf("Input transfer count values in decimal\n\n");
        printf("Read from address: ");
        gets(src);
        source=strtoul(src,NULL,16);
        printf("%lX\n",source);

        printf("Write to address: ");
        gets(dest);
        destination=strtoul(dest,NULL,16);
        printf("%lX\n",destination);

        printf("Read byte count: ");
        gets(rtc);
        readtc=strtoul(rtc,NULL,10);
        printf("%d\n",readtc);

        printf("Write byte count: ");
        gets(wtc);
        writetc=strtoul(wtc,NULL,10);
        printf("%d\n",writetc);

        outpd(op_reg_base_address+AMCC_OP_REG_MWAR, destination);
        outpd(op_reg_base_address+AMCC_OP_REG_MRAR, source);
        outpd(op_reg_base_address+AMCC_OP_REG_MWTC, writetc);
        outpd(op_reg_base_address+AMCC_OP_REG_MRTC, readtc);

        /* Enable Transfer Count interrupts */
        temp = inpd(op_reg_base_address+AMCC_OP_REG_INTCSR);
        outpd(op_reg_base_address+AMCC_OP_REG_INTCSR,temp|
                EN_READ_TC_INT|
                EN_WRITE_TC_INT);

        /* Enable Bus Mastering */
        temp = inpd(op_reg_base_address+AMCC_OP_REG_MCSR);
        outpd(op_reg_base_address+AMCC_OP_REG_MCSR, temp|RESET_A2P_FLAGS|
                RESET_P2A_FLAGS|
                A2P_HI_PRIORITY|
                P2A_HI_PRIORITY|
                EN_A2P_TRANSFERS|
                EN_P2A_TRANSFERS);
}
```

```
/*************************************************************************/
/*      Function:       handler                                          */
/*      Purpose:        Check interrupt source and service S5933 int's   */
/*      Inputs:         None                                             */
/*      Outputs:        None                                             */
/*************************************************************************/

void interrupt handler(void)
{
        byte    status;

        status = inportb(op_reg_base_address+AMCC_OP_REG_INTCSR+2);

        if((status & ANY_S5933_INT)!= 0){
                /* Disable bus mastering */
                outportb(op_reg_base_address+AMCC_OP_REG_MCSR+1,0x11);

                /* Identify AMCC Interrupt Source(s) */
                /* AMCC Hardware Interrupt Source */
                if((status & READ_TC_INT) != 0)
                        /* Read TC Interrupt Code Here */
                if((status & WRITE_TC_INT) != 0)
                        /* Write TC Interrupt Code Here */
                if((status & MASTER_ABORT_INT) != 0)
                        /* Master Abort Interrupt Code Here */
                if((status & TARGET_ABORT_INT) != 0)
                        /* Target Abort Interrupt Code Here */

                /* Clear Interrupt Enables */
                outportb(op_reg_base_address+AMCC_OP_REG_INTCSR+1,0);

                /* Clear all active interrupt sources */
                outportb(op_reg_base_address+AMCC_OP_REG_INTCSR+2,status);

                /* Reenable transfer count interrupts */
                outportb(op_reg_base_address+AMCC_OP_REG_INTCSR+1,0xC0);
        }
        else{
                /* Not an S5933 Interrupt Source */
                _chain_intr(oldhandler);
                }

        /* Specific End of interrupt to clear in-service bit(s)     */
        /* Mask upper 5 bits of int. line register                  */
        if(interrupt_line<8)
                outportb(0x20,0x60|(interrupt_line&0x07));
        else{
                /* Issue master then slave EOI */
                outportb(0xa0,0x60|((interrupt_line-8)&0x07));
                outportb(0x20,0x62);
                }
      }
}
```

```
/************************************************************************/
/* SETUP_INT_VECT     */
/*      */
/* Purpose:    Install the interrupt vector for the S5933 interrupt        */
/*      handler and reference the previous handler routine.         */
/* Inputs:     S5933 Operation registers base address       */
/* Outputs:    None                                                     */
/************************************************************************/

void setup_int_vect(byte bus_num,byte dev_func)

{
        byte    int_vector;
        int mask;

        if(read_configuration_byte(bus_num,dev_func,PCI_CS_INTERRUPT_LINE,
                &interrupt_line)==SUCCESSFUL){

                if(interrupt_line != 0xff){
                        if(interrupt_line<8){
                                int_vector=0x08+interrupt_line;
                        }
                        else{
                                int_vector=0x70+(interrupt_line-8);
                        }
                }

                /* Make sure the system 8259 enables the IRQ with OCW1 @ I/O 21h    */
                /* for IRQ0-7 or @ I/O A1h for IRQ8-15                              */
                if(interrupt_line < 8){
                        mask = inportb(0x21);
                        mask = mask & ~(1<<interrupt_line);
                        outportb(0x21,mask);
                        }
                else {
                        mask = inportb(0xA1);
                        mask = mask & ~(1<<(interrupt_line-8));
                        outportb(0xA1,mask);
                        }
        /* If AMCC interrupt service routine vector is not already installed,      */
        /* install it and save the old vector for chaining.                         */
        if(getvect(int_vector) != handler){
                /* Save the old interrupt vector */
                oldhandler=getvect(int_vector);

                /* Install the new interrupt vector */
                setvect(int_vector,handler);
                }
        }
}
```

INTRODUCTION

The S5933 allows PCI bus master transfers through the FIFO interface. The function of filling and emptying the FIFO is left to add-on logic. Many add-on designs implement a microprocessor or microcontroller with an integrated DMA controller that can perform this function. These devices can easily transfer data between the S5933 FIFO port and add-on memory.

Some add-on designs do not have processors or logic with DMA capabilities. This application note shows a programmable logic implementation of a simple, single channel DMA controller to perform add-on DMA transfers between the S5933 FIFO port and add-on memory. The design described allows simple implementation into an Intel 80960 processor-based add-on, but can easily be modified to support other add-on processors.

DMA CONTROLLER ARCHITECTURE

This DMA controller design has a single channel which can perform add-on reads and writes as a bus master on the add-on interface. An add-on transfer address and transfer byte count are programmed, and then DMA requests from the S5933 FIFO are monitored. Once a DMA request is received, the controller puts add-on logic in a hold state and when a hold acknowledge is returned, the DMA transfer begins.

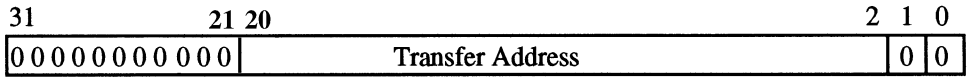
Register Architecture

There are two registers integrated in the DMA controller: a 22-bit address register and an 18-bit transfer count register. These registers are initialized by the processor before the DMA transfer begins. These registers are write-only. The DMA controller decodes address lines A21 and A20 on the add-on bus. This address decoding scheme can be easily modified to fit into the memory map of specific applications.

Address Register (ADDR)

The address register contains the address of the next DMA transfer. Before the DMA transfer begins, the ADDR register is written with the starting address of the DMA transfer. All transfers must take place on double-word boundaries (A1=A0=0). The ADDR register is incremented by 4 bytes after each data phase completes.

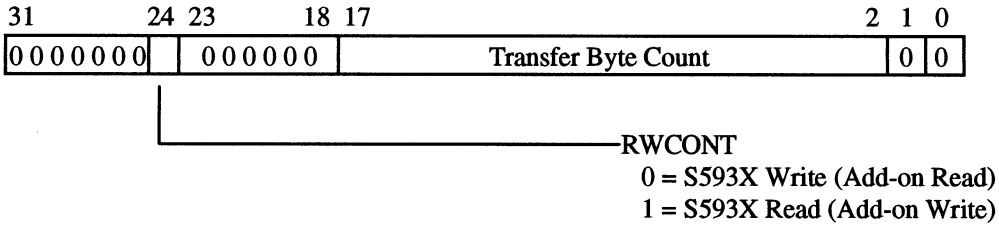
Register Name:	DMA Transfer Address (ADDR)
Location:	A21=0, A20=1
Power-up Value:	00000000h
Attribute:	Write Only
Size:	32-bits



Transfer Count Register (TXCNT)

The transfer count register contains the current number of bytes left to be transferred. Before the DMA transfer begins, the TXCNT register is written with the total number of bytes to be transferred. All transfers counts must be multiples of 4 bytes. The TXCNT register is decremented by 4 bytes after each data phase completes. Bit 24 of the TXCNT register controls the direction of the DMA transfer (read from the S5933 or write to the S5933).

Register Name: DMA Byte Transfer Count (TXCNT)
Location: A21=1, A20=1
Power-up Value: 00000000h
Attribute: Write Only
Size: 32-bits



DMA Controller Bus Interface

The DMA controller bus cycles are identical to bus cycles performed by an Intel 80960Jx processor. This allows the DMA controller to transfer data to and from any peripheral or memory controller compatible with the i960Jx processor. The PLD state machines can easily be modified to emulate other processor bus cycles.

The following signals make up the bus interface for the DMA controller:

Signal	Type	Function
A31:0	Bidirectional	Address/data lines for accessing DMA registers.
RDY_IN#	Input	Ready Input for DMA controller-generated bus cycles.
RDY_OUT#	Output	Ready Output for processor-generated bus cycles.
BLAST#	3-State	Indicates that the current data phase is the last data phase of a burst.
ADS#	Bidirectional	Indicates that valid address information is currently on the bus.
W/R#	Bidirectional	Write/Read indicator.
BE3:0#	3-State	Byte Enables. Always asserted for DMA transfers.
RDFIFO#	Output	Read one double-word from the S5933 FIFO.
WRFIFO#	Output	Write one double-word to the S5933 FIFO.
HOLD	Output	Add-on processor hold request.
HLDA	Input	Add-on processor hold acknowledge.
RESET#	Input	Add-on reset signal (from the S5933)

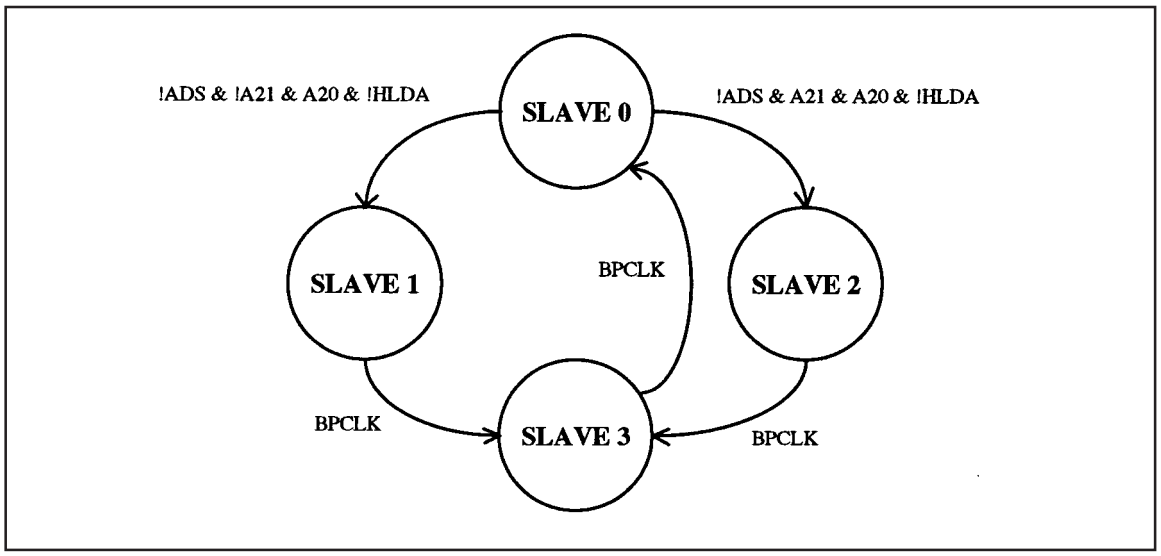
All bidirectional signals are driven by the current add-on bus master. When the add-on processor controls the bus, the DMA controller floats these outputs.

DMA Controller Slave Access State Machine

The DMA controller registers are mapped into add-on memory space. The address decode is a 2-bit decode of address bits A21 and A20. For add-on designs with conflicting memory requirements, the PLD equations can be easily modified to decode more address bits or different combinations of address bits.

When the controller decodes its address with ADS# asserted, a register access begins. HLDA (hold acknowledge) must be deasserted to allow a register access, indicating that the DMA controller is not the add-on bus master. The SLAVE state machine controls all accesses by the add-on processor or logic. Figure 1 shows the SLAVE state machine state diagram.

Figure 1. DMA Controller Slave State Machine

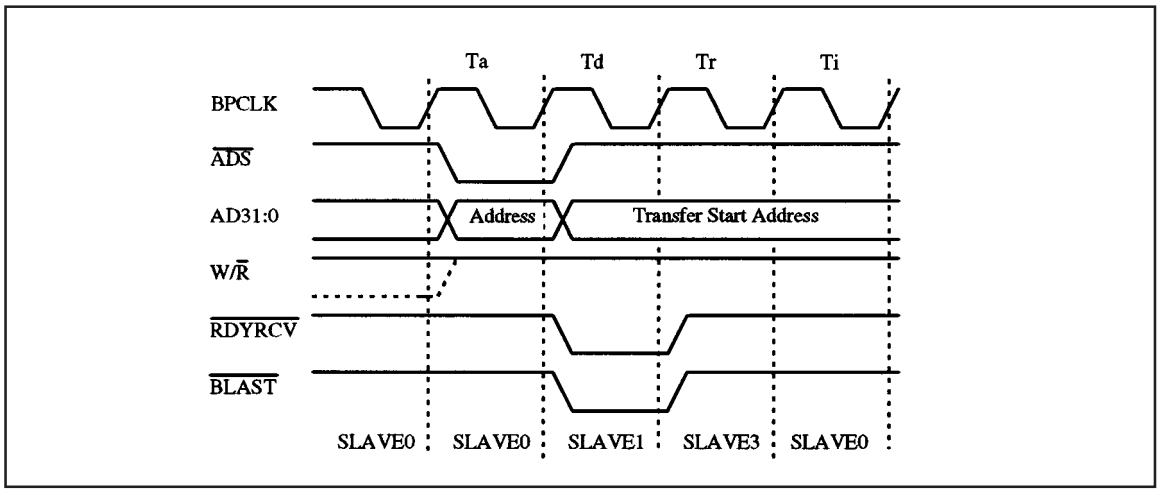


SLAVE 0 is the idle state. Depending on the state of A21 and A20 during the address phase of an access to the DMA controller, the SLAVE 1 or SLAVE 2 state is entered. SLAVE 1 is a write to the address register, SLAVE 2 is a write to the transfer count register. It only takes a single clock to write the register, so in SLAVE 1 and SLAVE 2, RDY_OUT# is asserted, and the next rising edge of BPCLK advances the state machine to SLAVE 3. SLAVE 3 is a recovery state (required for all 80960 bus cycle accesses). RDY_OUT# is deasserted, completing the access. The state machine returns to SLAVE 0 at the next rising edge of BPCLK.

For logic which does not require recovery states, SLAVE 3 may be removed, and the state machine can advance from SLAVE 1 or SLAVE 2 back to SLAVE 0 (the idle state).

Figure 2 shows a register write cycle by a 960Jx processor to the DMA controller. Ta indicates the processor address phase, Td is the data phase and Tr is the recovery phase. RDY_RCV# is an input into the 960 processor and is driven by the DMA controller RDY_OUT# signal. BLAST# is also driven by the processor and indicates the current data phase is the last data phase of a burst. For a single-cycle access, BLAST# is asserted during the first and only data phase.

Figure 2. Processor Write to DMA Address Register



DMA Controller Bus Master State Machine

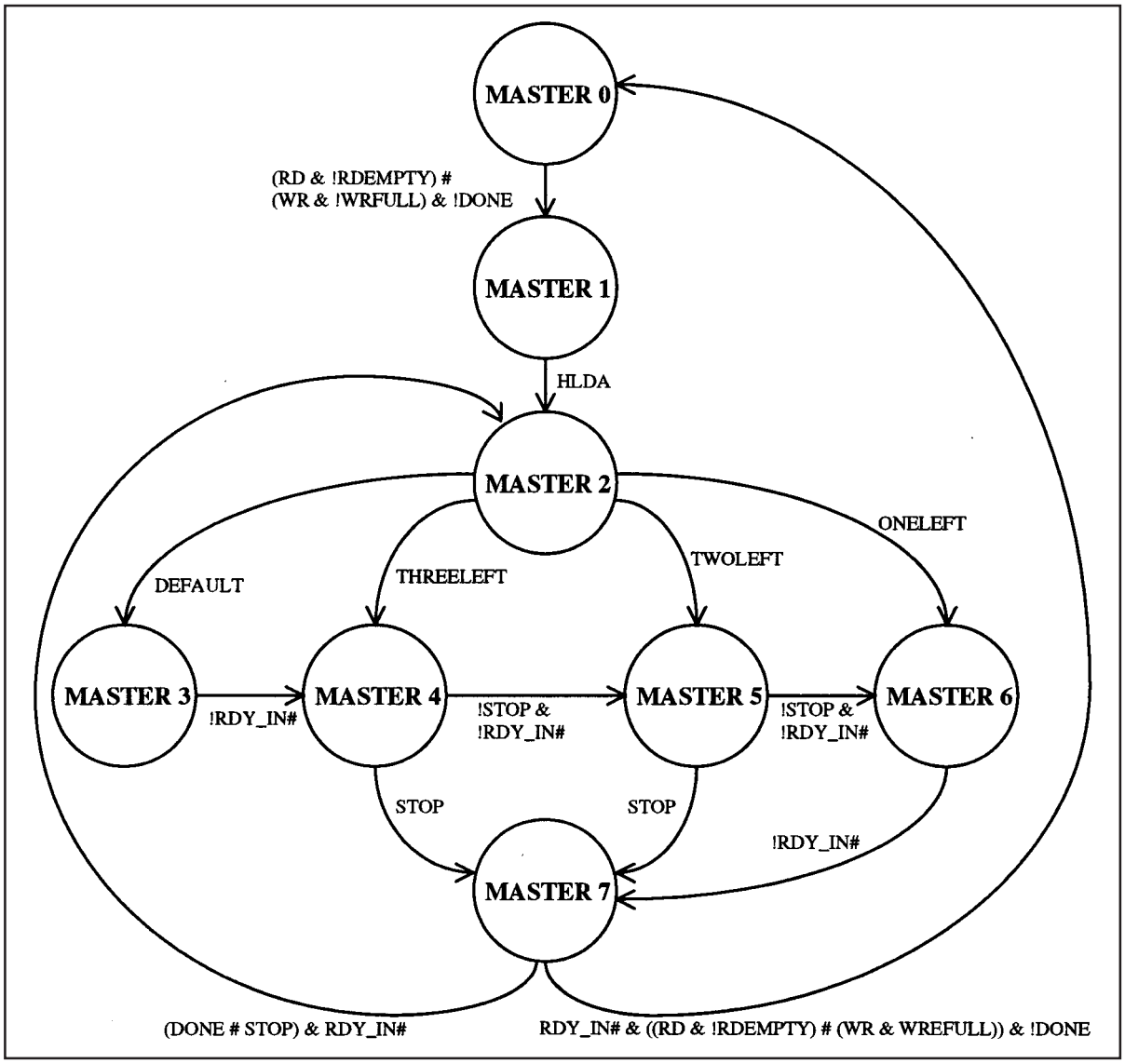
The MASTER state machine controls transfers across the add-on bus when DMA controller is the add-on bus master. In the idle state (MASTER 0), the controller monitors the S5933 FIFO status outputs. If the DMA controller is programmed to read from the PCI to add-on FIFO (RWCONT=1), RDEMPTY is monitored. If the DMA controller is programmed to write to the add-on to PCI FIFO (RWCONT=0), WRFULL is monitored. RDEMPTY and WRFULL act as the DMA requests (and they will be referred to as 'the DMA request' in the description of the MASTER state machine). Neither request will be acted upon by the DMA controller if the transfer count register is zero. Regardless of whether the transfer is to be a read or a write, the state machine operates the same way, only the W/R# output is different. The MASTER state machine state diagram is shown in Figure 3. For simplicity, only the conditions required to change states are shown. All other conditions result in the state machine remaining in the present state.

Once a valid DMA request is received (with a non-zero transfer count), the state machine advances to MASTER 1. In MASTER 1, HOLD is asserted to the add-on processor. The state machine does not advance to MASTER 2 until HLDA is returned by the processor, indicating that the DMA controller has access to the add-on bus. HLDA asserted enables the outputs for all of the bidirectional DMA controller signals (ADS#, BLAST#, W/R#, BE3:0#).

MASTER 2 is the address phase for an add-on bus cycle. In MASTER 2, ADS# is asserted and the outputs for the AD31:0 signals are enabled, driving the DMA transfer address on the add-on bus. The next rising edge of BPCLK advances the state machine to one of four data phases: MASTER 3-6. Which state occurs depends on the remaining transfer count value. Because BLAST# (last data phase of a burst indicator) is always asserted in MASTER 6, the state machine always operates so that the final operation of a DMA transfer ends in MASTER 6.

MASTER 3 is the first data phase of a four data phase burst. If the transfer count is 16 bytes or more in the MASTER 2 state, the state machine advances to MASTER 3 (because at least 4 more data phases are required). If the state machine enters MASTER 3, there is always data to transfer, so only RDY_IN# is monitored. The state machine remains in MASTER 3 until RDY_IN# is asserted by the add-on device being accessed. When RDY_IN# is asserted, the state machine advances to MASTER 4, and the ADDR and TXCNT registers are updated.

Figure 3. DMA Controller Add-on Bus Master State Machine



The MASTER 4 data phase can be entered from either the address phase (if the transfer count has decremented to 12) or from MASTER 3. During the MASTER 4 data phase, RDY_IN# and STOP are monitored. STOP is an internal signal which is asserted when a DMA request goes inactive. For example, if the last double-word in the PCI to add-on FIFO is read in MASTER 3, the RDEMPTY output is asserted. This causes STOP to be asserted in the MASTER 4 state. STOP asserted also prevents RDFIFO# and WRFIFO# from being asserted and does not allow data to be transferred. When STOP is assserted, the MASTER 4 state is completes when RDY_IN# is asserted, but the address and transfer count are not updated, and the state machine asserts BLAST# , advancing to MASTER 7 (recovery phase). If STOP is not asserted in MASTER 4, the data phase completes normally when RDY_IN# is asserted and the state machine advances to MASTER 5, updating the ADDR and TXCNT registers.

The MASTER 5 data phase is identical to MASTER 4. Master 5 can be entered from either the address phase (if the transfer count has decremented to 8) or from MASTER 4. If STOP is asserted, the state machine asserts BLAST# and advances to MASTER 7 when RDY_IN# is asserted. If STOP is not asserted, the state machine advances to MASTER 6 when RDY_IN# is asserted, updating the ADDR and TXCNT registers.

The MASTER 6 data phase can be entered from either the address phase (if the transfer count has decremented to 4) or from MASTER 5. Only RDY_IN# is monitored in MASTER 6. Because MASTER 7 is the next state anyway, there is no need to monitor STOP. BLAST# is always asserted during the MASTER 6 state. When RDY_IN# is asserted, the state machine advances to MASTER 7, and the ADDR and TXCNT registers are updated.

MASTER 7 is the recovery phase. In MASTER 7, RDY_IN#, RDEMPTY, WRFULL, STOP, and DONE are monitored. DONE is an internal signal which indicates that the transfer count is zero. Based on these signals, the state machine either returns to the idle state (MASTER 0) or starts another transfer by advancing to the address phase (MASTER 2). If RDY_IN# is low, the state machine remains in MASTER 7. If RDY_IN# is high and the DMA request is still active (RDEMPTY or WRFULL deasserted), the state machine advances to MASTER 2 to begin another data transfer. If RDY_IN# is high and DONE or STOP is asserted (either the transfer count is zero, or the FIFO cannot support another transfer yet), then the state machine returns to MASTER 0 and relinquishes control of the add-on bus by deasserting HOLD.

DMA Controller Bus Operation

The DMA controller emulates bus cycles of an Intel 960Jx processor. This allows the controller to transfer data directly to and from memory controllers or peripherals designed for the 960. The S5933 FIFO allows an unlimited burst length, but the 960 supports a maximum of 4 data phases burst accesses. Therefore, the DMA controller bursts no more than four consecutive data phases before issuing an address phases.

FIFO DMA Request for Writes to an Add-on Destination

For the S5933 PCI to add-on FIFO, RDEMPTY is deasserted when there is valid data in the FIFO. The FIFO can be filled by another PCI bus initiator using the S5933 as a target, or the S5933 can fill its own FIFO by acting as a PCI bus master. The DMA controller functions identically in either case. RDEMPTY deasserted acts as a DMA request to the controller when RWCONT =1.

FIFO DMA Request for Reads from an Add-on Source

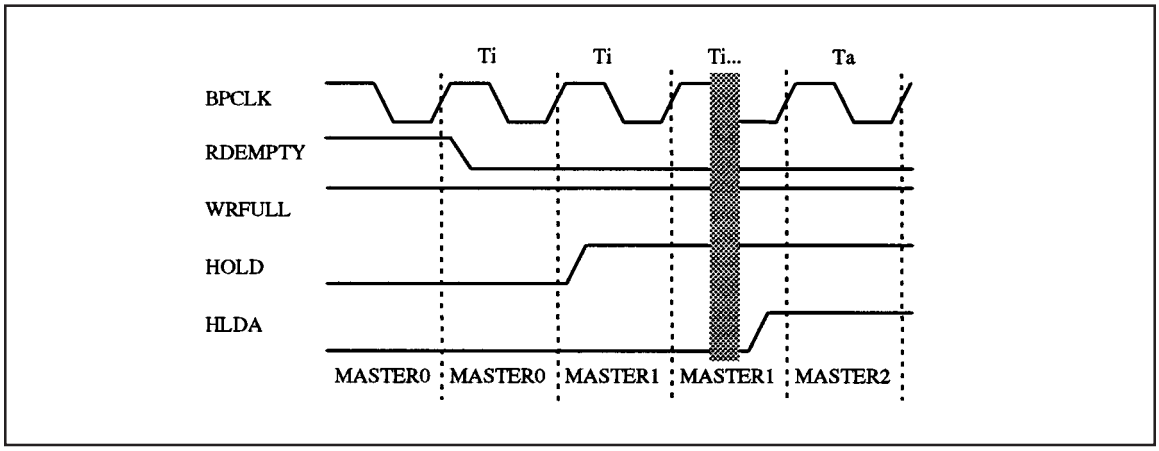
For the S5933 add-on to PCI FIFO, WRFULL is deasserted when the add-on to PCI FIFO is completely full. The FIFO can be emptied by another PCI bus initiator using the S5933 as a target, or the S5933 can empty its own FIFO by acting as a PCI bus master. The DMA controller functions identically in either case. WRFULL deasserted acts as a DMA request to the controller when RWCONT=0.

DMA Requests and Add-on Bus Arbitration

When the DMA controller has been programmed with a source address (for DMA reads) or destination address (for DMA writes) and a transfer byte count, it is ready to receive DMA requests from the S5933. For DMA reads (RWCONT=1), RDEMPTY acts as the request. Whenever RDEMPTY is low, there is data in the PCI to add-on FIFO that needs to be transferred to an add-on destination. For DMA writes (RWCONT=0), WRFULL acts as the request. Whenever WRFULL is low there are empty locations in the add-on to PCI FIFO that need to be filled from an add-on source. RDEMPTY or WRFULL low starts the MASTER state machine.

Figure 4 shows the sequence for the DMA controller becoming the add-on bus master. The DMA controller becomes an add-on bus master using the 960 processor's hold/hold acknowledge protocol. When a DMA request is received, HOLD is asserted. When the processor returns HLDA, the DMA controller can begin transferring data to or from the S5933 FIFO. The DMA controller remains bus master until the request is removed (RDEMPTY or WRFULL asserted).

Figure 4. DMA Request and Add-on Bus Arbitration

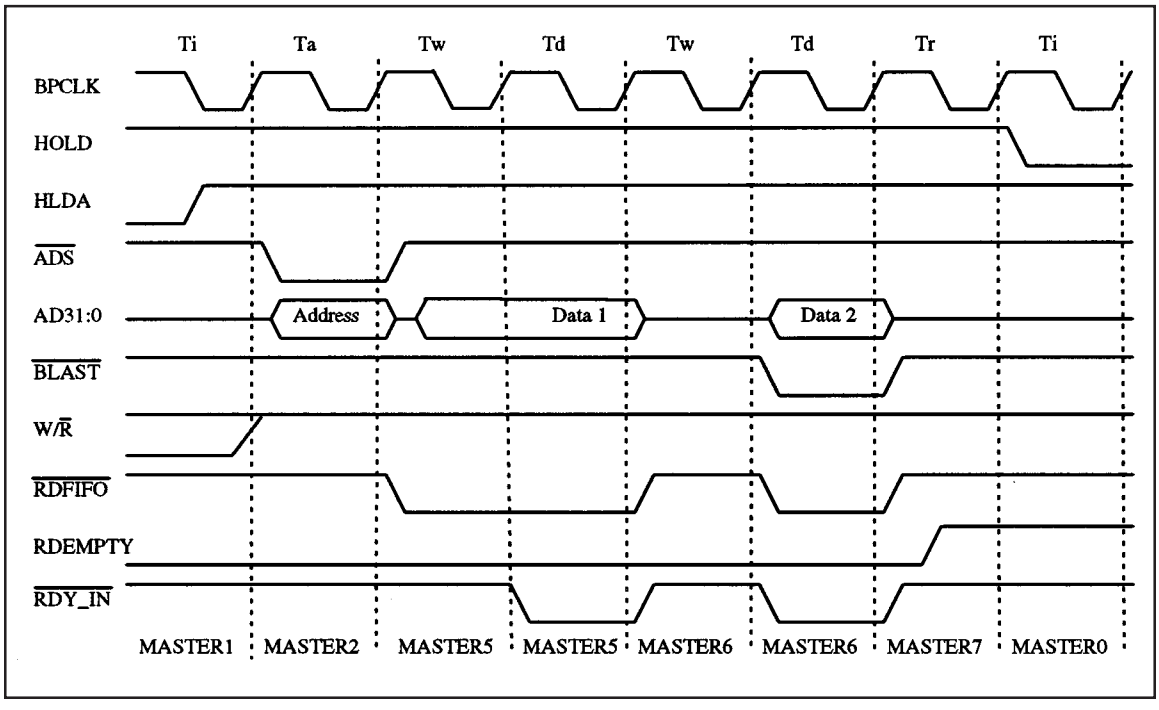


DMA Burst Reads

The DMA controller always attempts to read from the S5933 PCI to add-on FIFO in bursts. Figure 5 shows a two data phase burst of the last two double words in a transfer (TXCNT = 8). Typically, a write to DRAM has more wait states on the first data phase, but this is not shown, for simplicity. RDY_IN# is driven by an external device such as a memory controller and may be used to extend individual data phases for as long as necessary.

The DMA controller is designed to interface with the FIFO asynchronous to BPCLK. This means that bit 6 of location 45h of the non-volatile boot device for the S5933 should be set. In this mode of operation, the S5933 advances the FIFO pointer at each rising edge of RDFIFO#. To interface to a zero wait-state device, the S5933 would have to be configured for FIFO operation synchronous to BPCLK (where the FIFO advances at every BPCLK rising edge when RDFIFO# is asserted). This would also require modification of the PLD equations.

Figure 5. DMA Burst Read from the S5933 FIFO (Add-on Write)

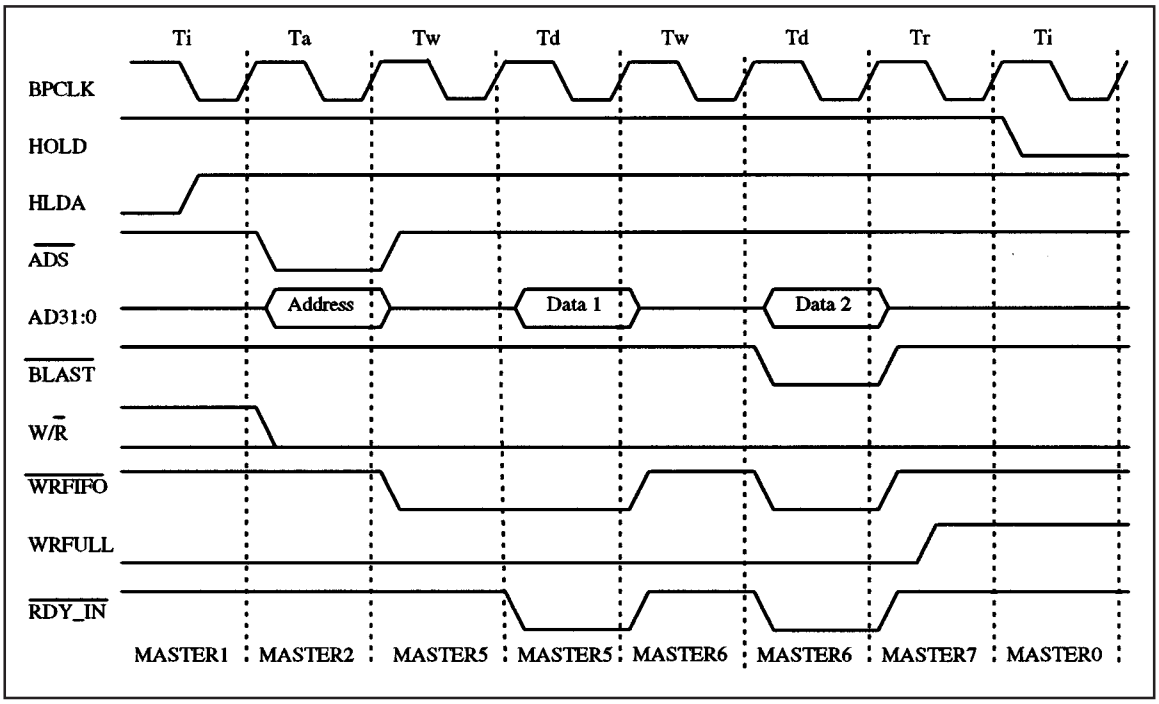


DMA Burst Writes

The DMA controller always attempts to write to the S5933 add-on to PCI FIFO in bursts. Figure 6 shows a two data phase burst of the last two double words in a transfer (TXCNT = 8). RDY_IN# is driven by an external device such as a memory controller and may be used to extend individual data phases for as long as necessary.

The DMA controller is designed to interface with the FIFO asynchronous to BPCLK. This means that bit 5 of location 45h of the non-volatile boot device for the S5933 should be set. In this mode of operation, the S5933 advances the FIFO pointer at each rising edge of WRFIFO#. To interface to a zero wait-state device, the S5933 would have to be configured for FIFO operation synchronous to BPCLK (where the FIFO advances at every BPCLK rising edge when WRFIFO# is asserted). This would also require modification of the PLD equations.

Figure 6. DMA Burst Write to the S5933 FIFO (Add-on Read)



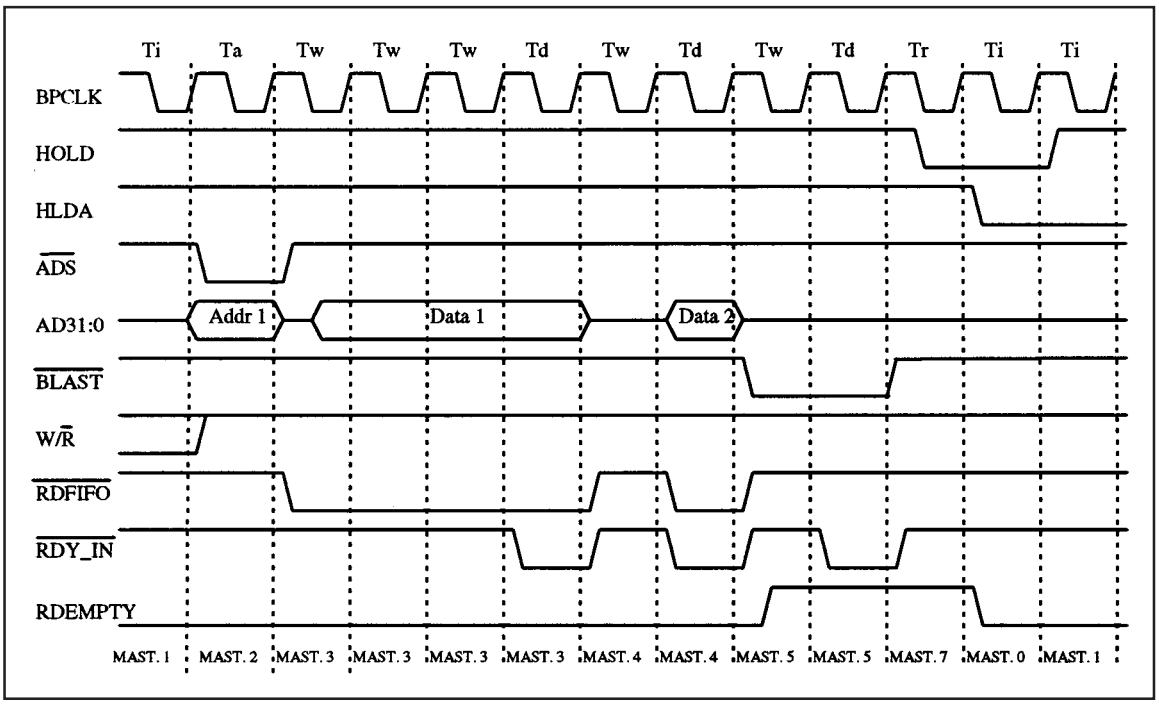
Removal of a DMA Request During a Burst

It is possible that during a burst read or write operation by the DMA controller, the DMA request will be removed. This indicates that the S5933 FIFO cannot support any more data transfers. For DMA reads from the S5933 PCI to add-on FIFO, this occurs when RDEMPTY is asserted. This indicates that there is no data in the FIFO to be read. For DMA writes to the S5933 add-on to PCI FIFO, this occurs when WRFULL is asserted. This indicates that the FIFO is full and no more data should be written.

The DMA controller state machine provides for this situation. Figure 7 shows a DMA burst read from the S5933 FIFO. Data 1 is transferred to Address 1, Data 2 is transferred to Address 1 + 4. RDEMPTY is asserted after the second double-word is transferred, indicating there is currently no more data to be read. The state machine has already advanced to the next data phase, but when RDEMPTY is asserted, RDFIFO# is not asserted during the final data phase, and the address and transfer count registers are not modified. This prevents the FIFO pointers from being updated, but a dummy write cycle is still run to memory at Address 1 + 8. When RDEMPTY is deasserted again (indicating there is more data available), the DMA transfer continues from the next address (Address 1 +8). This results in the data written by the dummy cycle to be overwritten with the next valid data.

When a DMA request is removed during a data phase, the DMA controller completes the data phase (without asserting RDFIFO# or WRFIFO# or updating the address/count registers) and advances to the recovery phase. From the recovery phase, if the DMA request has not been reasserted, the state machine advances to the idle state, giving up control of the add-on bus. If the DMA request has been reasserted by the time the recovery phase is completed, then the state machine advances to the address phase and begins another data transfer.

Figure 7. DMA Request (RDEMPTY) Removal During a Burst Read Transfer



PLD MODIFICATIONS

The PLD implementation described in this application note is a general-purpose, single-channel DMA controller. The design can easily be modified for a specific add-on application. Modifications such as decreasing the address register or transfer count size are relatively simple. The design could also be easily modified for a 16-bit add-on interface. The following sections describe some possible modifications and show how they can be implemented.

Automatically Programming Transfer Count and Address Registers

Many S5933 applications may implement add-on initiated bus mastering. This allows add-on logic to program S5933 PCI bus master address registers (MWAR and MRAR) and transfer count registers (MWTC and MRTC). In this situation, it might be desirable to allow the DMA controller address and transfer count registers to be programmed during the write access to the corresponding S5933 add-on operation registers. Whenever a PCI bus master read or write address is programmed, the DMA controller address register is programmed during the same cycle. Whenever a PCI bus master read or write transfer count is programmed, the DMA controller transfer count is programmed during the same cycle. This avoids running extra add-on bus cycles to setup add-on DMA transfers and S5933 bus mastering.

To do this, the PLD must decode the addresses for the S5933 bus master address and transfer count registers. The following S5933 add-on operation register locations must be decoded:

Register	Offset
Bus Master Read Address Register (MRAR)	30h
Bus Master Write Address Register (MWAR)	24h
Bus Master Read Transfer Count (MRTC)	5Ch
Bus Master Write Transfer Count (MWTC)	58h

Add-on bus address lines A6:2 must be decoded. An access to either MRAR or MWAR programs the DMA address register. It should be noted that MWAR and MRAR are 32-bit registers (as PCI memory space is 4 GB), only the lower 22 bits are stored by the DMA controller address register (which should be sufficient for most applications).

An access to either MRTC or MWTC programs the DMA transfer count register. Bit 24 of the DMA controller is the RWCONT bit (indicating a read or write from the S5933). This bit should be programmed based on which transfer count register of the S5933 is programmed (indicated by A2).

The following PLD equation modifications are required to implement this function:

```
/* Inputs */
Pin   =     nSELECT;    /* S5933 SELECT# Input */

/* Logic Equations */

/* For MWTC and MRTC accesses, A2 differentiates between PCI read and write */
/* transfer count registers */

RWCONT.T = LOADC & (RWCONT $ A2.IO);

SEQUENCE slave {
/* Idle State */
present S0
      if !nADS.IO & !nSELECT & !HLDA &
            ((A5.IO & A4.IO & !A3.IO & !A2.IO) #
             (A5.IO & !A4.IO & !A3.IO & A2.IO)) next S1;
      if !nADS.IO & !nSELECT & !HLDA & A5.IO    next S2;
      default next S0;

present S1        /* Address Access Data Phase */
      next S3;

present S2        /* Transfer Count Access Data Phase */
      next S3;

present S3        /* Recovery Phase */
      next S0;
}
```

Controlling S5933 PCI Bus Mastering with AMREN and AMWEN

For S5933 applications implementing add-on initiated bus mastering, the AMREN and AMWEN inputs enable the S5933 to become a PCI bus master. These signals can be sourced from the DMA controller and asserted when the controller is ready to service DMA requests. If add-on transfer counts are enabled for the S5933, this function is not required, as the AMREN and AMWEN may remain asserted and PCI bus mastering is internally enabled by a non-zero transfer count value.

AMREN should be asserted for DMA S5933 reads (RWCONT=1) and AMWEN should be asserted for DMA S5933 writes (RWCONT=0). Once the DMA controller is initialized (address and transfer count written), the appropriate enable should be asserted. When the transfer count reaches zero, the enable must be deasserted, and the FIFO pointers should be reset (using FRC# for the PCI to add-on FIFO and FWC# for the add-on to PCI FIFO). It is only really necessary to reset the PCI to add-on FIFO because extra data may have been read which is not required. This must be flushed from the FIFO. Extra data does not enter the add-on to PCI FIFO without WRFIFO# being asserted, the DMA controller logic prevents this.

The following PLD equation modifications are required to implement this function:

```
/* Inputs */
Pin   =     FWE;        /* Add-on to PCI FIFO Empty indicator */

/* Outputs */

Pin   =     nFRC;       /* PCI to add-on FIFO reset */
Pin   =     AMREN;           /* PCI to add-on FIFO PCI master enable */
Pin   =     AMWEN;           /* Add-on to PCI FIFO PCI master enable */

/* Logic Equations */

AMREN = !DONE & RD;          /* Assert for reads with tc != zero */
AMWEN = (!DONE # !FWE) & WR;  /* Assert for writes with tc != zero and */
                        /* FIFO not empty. The transfer count can */
                        /* decrement to zero, but bus master writes */
                        /* still remain, FWE indicates when the FIFO */
                        /* has finished all transfers */

nFRC = !(DONE & MASTER7 & RD); /* Assert the PCI to add-on FIFO reset */
                         /* during the recovery phase when the */
                         /* transfer count reaches zero */
```

Programming the DMA Controller with the Pass-thru Interface

Some applications may not implement a processor or intelligent logic on the add-on interface. This requires an alternate method to program the DMA controller address and transfer count registers. The S5933 pass-thru interface provides a simple method to perform this task with the host CPU. A Base Address Register is required to define a pass-thru region for the DMA registers. An 8 byte I/O region is all that is needed (a decode of the S5933 PTNUM1:0 outputs and pass-thru address line A2 are used for the add-on address decode).

Not implementing an add-on processor changes a number of things in the PLD. HOLD and HLDA are no longer required, as the DMA controller is always add-on bus master in this implementation. RDY_OUT# is no longer required (as it acted as a ready indicator to the 960 processor). RDY_OUT# is replaces with PTRDY#, which indicates a pass-thru access to a DMA controller register is now complete. PTATN# acts as a request for the DMA controller to read the pass-thru data register (writing the address or transfer count register).

The DMA controller MASTER state machine is modified slightly. The MASTER 1 state changes to prevent the controller from starting an add-on DMA transfer while a pass-thru access to one of its registers is occurring. The SLAVE machine requires more modification. SLAVE 0 only monitors PTATN#. If PTATN# is asserted and a DMA transfer is not in progress, the state machine advances to SLAVE 1. During SLAVE 1, PTADR# is asserted and the pass-thru address is decoded, determining if the DMA address or transfer count is to be written (or if the pass-thru access is not intended for the DMA controller). SLAVE 2 and SLAVE 3 program the appropriate DMA registers, asserting RD# and PTRDY#. The state machine then advances to SLAVE 0, returning the machine to its idle state.

The following PLD equation modifications are required to implement this function:

```
/* Inputs */
Pin   =     nPTATN;           /* Pass-thru Access Indicator */
Pin   =     PTNUM0;           /* Pass-thru Region Decode */
Pin   =     PTNUM1;
                              /* Remove HLDA Output */

/* Outputs */

Pin   =     nPTADR;           /* Pass-thru Address Strobe */
Pin   =     nRD;              /* S5933 RD# Input */
Pin   =     nPTRDY;           /* Pass-thru Access Complete Indicator */
                              /* ADR6:2 are hardwired to the PTDATA address */
                              /* HOLD output is no longer required */
                              /* nRDY_OUT becomes nPTRDY */
                              /* SELECT# and BE3:0# on the S5933 may be tied low */

/* Declarations and Intermediate Variable Definitions */

/* Remove all equations for HOLD, HLDA, nRDY_OUT */

BE3.OE      = 'b'1;           /* DMA controller is only add-on bus master */
BE2.OE      = 'b'1;
BE1.OE      = 'b'1;
BE0.OE      = 'b'1;
nPTRDY.OE = 'b'1;
nADS.OE     = 'b'1;
W_nR.OE     = 'b'1;


/* Logic Equations */

/* The bus add-on bus master state machine and the add-on slave */
/* state machine change to support pass-thru programming (pt-writes) and */
/* no add-on processor */
/* Remove all equations for HOLD, HLDA, nRDY_OUT */

nPTADR = !SLAVE1;
nRD = !(SLAVE2 # SLAVE3);
nPTRDY = !(SLAVE2 # SLAVE3);
```

```
SEQUENCE master {

present M0
      if ((RD & !RDEMPTY) # (WR & !WRFULL)) & !DONE next M1;

present M1  /* THIS IS THE ONLY STATE WHICH CHANGES */
      if !SLAVE0 next M1;     /* Do not interrupt a controller register access */
      if SLAVE0 next M2;      /* No register access, so continue */
   •
   •
   •
}

SEQUENCE slave {

present S0        /* Idle State */
      /* If the pass-thru access is occurring and */
      /* the master state machine is in the idle */
      /* state, begin the sequence. */

      if !nPTATN & MASTER0 next S1;
      if nPTATN # !MASTER0 next S0;

present S1        /* PTADR# Address Phase */
      /* PTADR# is asserted in this phase. Address decode assumes */
      /* an 8 byte region is defined for pass-thru (BADR4), and only A2 is decoded */

      if PTNUM0 & PTNUM1 & A2 next S2;    /* Address register write */
      if PTNUM0 & PTNUM1 & !A2      next S3;    /* Transfer count write */
      default next S0;

present S2        /* Address Access Data Phase - Assert PTRDY# and RD# */
      next S0;

present S3        /* Transfer Count Access Data Phase - Assert PTRDY# and RD#*/
      next S0;

/* No recovery phase is required, as the S5933 does not require it */
}
```

PLD EQUATIONS

The following code is written in CUPL™ by Logical Devices. Pin assignments will depend on the programmable logic device the code is programmed into. The code compiles and simulates without errors.

```
Name      DMA_CON;
Partno     ;
Date       5/19/95;
Revision   0;
Designer   JMW;
Company    AMCC;
Assembly    ;
Location    ;
Device      ;

/*****************************************************************/
/* Add-on DMA Controller                                         */
/* The PLD equations below compile and have been simulated and are*/
/* believed to be correct.  AMCC assumes no liability for errors  */
/* made in the code or logic                                     */
/*                                                               */
/*                                                               */
/*****************************************************************/
/*  Allowable Target Device Types:                               */
/*****************************************************************/

/**  Inputs  **/

Pin          = BPCLK     ;      /* Buffered PCI Clock                */
Pin          = RDEMPTY   ;      /* PCI to Add-on FIFO Empty Indicator */
Pin          = WRFULL    ;      /* Add-on to PCI FIFO Full Indicator  */
Pin          = HLDA      ;      /* CPU Hold Acknowledge              */
Pin          = [A7..0]   ;      /* Address/Data Bus 7:0          */
Pin          = [A15..8]  ;      /* Address/Data Bus 15:8         */
Pin          = [A23..16] ;      /* Address/Data Bus 23:16        */
Pin          = [A31..24] ;      /* Address/Data Bus 31:24        */
Pin          = nRDY_IN   ;      /* DMA Controller Ready Input    */
Pin          = nRESET    ;      /* S5933 Reset Output            */

/**  Outputs  **/

Pin          = HOLD      ;      /* CPU Hold                      */
Pin          = W_nR      ;      /* Bidirectional WR/RD           */
Pin          = nRDY_OUT  ;      /* Ready Output (to CPU)         */
Pin          = nBLAST    ;      /* Last Data Phase of Burst Indicator */
Pin          = nADS      ;      /* Bidirectional Address Strobe   */
Pin          = nRDFIFO   ;      /* S5933 FIFO Read Input         */
Pin          = nWRFIFO   ;      /* S5933 FIFO Write Input        */
Pin          = BE3       ;      /* Byte Enable Output 3          */
Pin          = BE2       ;      /* Byte Enable Output 2          */
Pin          = BE1       ;      /* Byte Enable Output 1          */
Pin          = BE0       ;      /* Byte Enable Output 0          */
```

```
/** Declarations and Intermediate Variable Definitions **/

/* State machine bits are defined as nodes.  The MA bits are the  */
/* MASTER state machine and the SL bits are the SLAVE state */
/* machine.  The 18-bit TXCNT register is composed of C17:2.  It  */
/* is divided into two groups to allow a easier fit into some     */
/* FPGAs. RWCONT is an additional bit in TXCNT which indicates if */
/* the DMA transfer is to be a S5933 read or write.  nREADY is a  */
/* registered version of nRDY_IN used in the logic for generating */
/* nRDFIFO and nWRFIFO.                                           */

NODE [C7..2];           /* TXCNT Bits 2-7   */
NODE [C17..8];          /* TXCNT Bits 8-15  */
NODE [MA2..0];          /* MASTER State Machine Bits */
NODE [SL1..0];          /* SLAVE State Machine Bits */
NODE RWCONT;            /* DMA Read/Write Indicator */
NODE nREADY;            /* Registered nRDY_IN Signal */

[C7..2].AR = !nRESET;   /* Reset all registers/state machines to zero */
[C17..8].AR = !nRESET;
[MA2..0].AR = !nRESET;
[SL1..0].AR = !nRESET;
RWCONT.AR = !nRESET;
nREADY.AR='b'0;

[C7..2].AP ='b'0;
[C17..8].AP ='b'0;
RWCONT.AP ='b'0;
[MA2..0].AP = 'b'0;
[SL1..0].AP = 'b'0;
nREADY.AP=!nRESET;

[C7..2].CK = BPCLK;     /* All reg's/state machines clocked w/BPCLK */
[C17..8].CK = BPCLK;
nREADY.CK = BPCLK;
[SL1..0].CK = BPCLK;
[MA2..0].CK = BPCLK;
[A7..0].CK = BPCLK;
[A15..8].CK = BPCLK;
[A23..16].CK = BPCLK;
[A31..24].CK = BPCLK;
BE3.CK = BPCLK;
BE2.CK = BPCLK;
BE1.CK = BPCLK;
BE0.CK = BPCLK;
RWCONT.CK =BPCLK;
```

```
BE3.OE = HLDA;          /* HLDA indicates that the DMA controller may */
BE2.OE = HLDA;          /* become bus master.  All outputs are enabled*/
BE1.OE = HLDA;          /* when HLDA is asserted */
BE0.OE = HLDA;
nRDY_OUT.OE = HLDA;
nBLAST.OE = HLDA;
nADS.OE = HLDA;
[A7..0].OE = MASTER2;   /* Address outputs only enabled during MASTER2*/
[A15..8].OE = MASTER2;  /* which is the address phase of the transfer */
[A23..16].OE = MASTER2;
[A31..24].OE = MASTER2;
W_nR.OE = HLDA;

/* Shorthand notation for address and transfer count values */

BYTE0 = A7 & A6 & A5 & A4 & A3 & A2;
BYTE1 = A15 & A14 & A13 & A12 & A11 & A10 & A9 & A8;
BYTE2 = A13 & A22 & A21 & A20 & A19 & A18 & A17 & A16;
CBYTE0 = !C7 & !C6 & !C5 & !C4 & !C3 & !C2;
CBYTE1 = !C17 & !C16 & !C15 & !C14 & !C13 & !C12 & !C11 & !C10 & !C9 & !C8;

/* Need to know when count is below 4 data phases (double-words) for */
/* the MASTER state machine */

ONELEFT = CBYTE1 & !C7 & !C6 & !C5 & !C4 & !C3 & C2;
TWOLEFT = CBYTE1 & !C7 & !C6 & !C5 & !C4 &  C3 & !C2;
THREELEFT = CBYTE1 & !C7 & !C6 & !C5 & !C4 & C3 & C2;

/* A read indicates an S5933 read and an add-on Write.  A write */
/* indicates an S5933 write and an add-on read                  */

RD = RWCONT;
WR = !RWCONT;

/* STOP indicates a condition where the FIFO cannot support     */
/* further transfers (empty or full) and the current burst must */
/* terminate.                                                   */

STOP = (RD & RDEMPTY) # (WR & WRFULL);

/* DONE indicates the transfer count is zero and transfers should */
/* stop.                                                           */

DONE = CBYTE0 & CBYTE1;

/* Increment address and decrement TC after each completed data   */
/* phase.  Do not count if the read FIFO is empty or the write    */
/* FIFO is full.  Run a dummy cycle and begin again when the      */
/* transfer can continue.  This assumes a 1 clock RDY pulse.  CNT */
/* only toggles when the DMA controller is local bus master */
/* (HLDA=1).                                                      */
```

```
CNT = !nRDY_IN & !STOP & HLDA & nRESET &
      (MASTER3 # MASTER4 # MASTER5 # MASTER6);

LOADA =  W_nR.IO & SLAVE1;    /* Load Address Register           */
LOADC =  W_nR.IO & SLAVE2;    /* Load Transfer Count Register    */


/**  Logic Equations  **/

nREADY.D = nRDY_IN;           /* Used for RDFIFO#/WRFIFO# Timing  */

[BE3..0] = 'b'0000;           /* All Add-on accesses are 32-bits  */
[A31..25] = 'b'0000000;       /* A31..22, A1..0 Driven to zero    */
A24 = A24.IO & !HLDA;
[A23..22] = 'b'00;
[A1..0] = 'b'00;

/* Toggle output when CNT is asserted and all previous bits are  */
/*'1' or when a LOAD is performed and the load value (.IO)       */
/* is different from the existing value.                         */

Field ADDR_COUNTA = [A7..2];

A2.T = CNT # LOADA & (A2 $ A2.IO);
A3.T = CNT & A2 # LOADA & (A3 $ A3.IO);
A4.T = CNT & A2 & A3 # LOADA & (A4 $ A4.IO);
A5.T = CNT & A2 & A3 & A4 # LOADA & (A5 $ A5.IO);
A6.T = CNT & A2 & A3 & A4 & A5 # LOADA & (A6 $ A6.IO);
A7.T = CNT & A2 & A3 & A4 & A5 & A6 # LOADA & (A7 $ A7.IO);

Field ADDR_COUNTB = [A15..8];

A8.T = CNTB2 # LOADA & (A8 $ A8.IO);
A9.T = CNTB2 & A8 # LOADA & (A9 $ A9.IO);
A10.T = CNTB2 & A8 & A9 # LOADA & (A10 $ A10.IO);
A11.T = CNTB2 & A8 & A9 & A10 # LOADA & (A11 $ A11.IO);
A12.T = CNTB2 & A8 & A9 & A10 & A11 # LOADA & (A12 $ A12.IO);
A13.T = CNTB2 & A8 & A9 & A10 & A11 & A12 # LOADA & (A13 $ A13.IO);
A14.T = CNTB2 & A8 & A9 & A10 & A11 & A12 & A13 # LOADA & (A14 $ A14.IO);
A15.T = CNTB2 & A8 & A9 & A10 & A11 & A12 & A13 & A14 # LOADA & (A15 $ A15.IO);

/* Assert CNTB2 when CNT is asserted and A2-7 are set */

CNTB2 = CNT & (A2 & A3 & A4 & A5 & A6 & A7);

Field ADDR_COUNTC = [A21..16];

A16.T = CNTC2 # LOADA & (A16 $ A16.IO);
A17.T = CNTC2 & A16 # LOADA & (A17 $ A17.IO);
A18.T = CNTC2 & A16 & A17 # LOADA & (A18 $ A18.IO);
A19.T = CNTC2 & A16 & A17 & A18 # LOADA & (A19 $ A19.IO);
A20.T = CNTC2 & A16 & A17 & A18 & A19 # LOADA & (A20 $ A20.IO);
A21.T = CNTC2 & A16 & A17 & A18 & A19 & A20 # LOADA & (A21 $ A21.IO);
```

```
/* Assert CNTC2 when CNTB2 is asserted and A8-15 are set */

CNTC2 = CNTB2 & (A8 & A9 & A10 & A11 & A12 & A13 & A14 & A15);

/* Toggle output when CNT is asserted and all previous bits are   */
/* '1' or when a LOAD is performed and the load value (.IO) is    */
/* different from the existing value. The transfer count is a     */
/* byte count.                                                    */

Field XFER_COUNTA = [C7..2];

C2.T = CNT # LOADC & (C2 $ A2.IO);
C3.T = CNT & !C2 # LOADC & (C3 $ A3.IO);
C4.T = CNT & !C2 & !C3 # LOADC & (C4 $ A4.IO);
C5.T = CNT & !C2 & !C3 & !C4 # LOADC & (C5 $ A5.IO);
C6.T = CNT & !C2 & !C3 & !C4 & !C5 # LOADC & (C6 $ A6.IO);
C7.T = CNT & !C2 & !C3 & !C4 & !C5 & !C6 # LOADC & (C7 $ A7.IO);

/* Assert CNTA when CNT is asserted and C2-7 are clear */

CNTA = CNT & !(C2#C3#C4#C5#C6#C7);

Field XFER_COUNTB = [C17..8];

C8.T = CNTA # LOADC & (C8 $ A8.IO);
C9.T = CNTA & !C8 # LOADC & (C9 $ A9.IO);
C10.T = CNTA & !C8 & !C9 # LOADC & (C10 $ A10.IO);
C11.T = CNTA & !C8 & !C9 & !C10 # LOADC & (C11 $ A11.IO);
C12.T = CNTA & !C8 & !C9 & !C10 & !C11 # LOADC & (C12 $ A12.IO);
C13.T = CNTA & !C8 & !C9 & !C10 & !C11 & !C12 # LOADC & (C13 $ A13.IO);
C14.T = CNTA & !C8 & !C9 & !C10 & !C11 & !C12 & !C13 # LOADC & (C14 $ A14.IO);
C15.T = CNTA & !C8 & !C9 & !C10 & !C11 & !C12 & !C13 & !C14 # LOADC & (C15 $
A15.IO);
C16.T = CNTA & !C8 & !C9 & !C10 & !C11 & !C12 & !C13 & !C14 & !C15 # LOADC & (C16 $
A16.IO);
C17.T = CNTA & !C8 & !C9 & !C10 & !C11 & !C12 & !C13 & !C14 & !C15 & !C16 # LOADC &
(C17 $ A17.IO);


/* This bit identifies a read vs. a write DMA 1 = read, 0 = write */

RWCONT.T = LOADC & (RWCONT $ A24.IO);

FIELD master = [MA2..0];
$DEFINE M0      'b'000
$DEFINE M1      'b'001
$DEFINE M2      'b'011
$DEFINE M3      'b'010
$DEFINE M4      'b'110
$DEFINE M5      'b'111
$DEFINE M6      'b'101
$DEFINE M7      'b'100
```

```
MASTER0 = !MA2 & !MA1 & !MA0        ;
MASTER1 = !MA2 & !MA1 &  MA0        ;
MASTER2 = !MA2 &  MA1 &  MA0        ;
MASTER3 = !MA2 &  MA1 & !MA0        ;
MASTER4 =  MA2 &  MA1 & !MA0        ;
MASTER5 =  MA2 &  MA1 &  MA0        ;
MASTER6 =  MA2 & !MA1 &  MA0        ;
MASTER7 =  MA2 & !MA1 & !MA0        ;

FIELD slave = [SL1..0];
$DEFINE S0      'b'00
$DEFINE S1      'b'01
$DEFINE S2      'b'10
$DEFINE S3      'b'11

SLAVE0 = !SL1 & !SL0        ;
SLAVE1 = !SL1 &  SL0        ;
SLAVE2 =  SL1 & !SL0        ;
SLAVE3 =  SL1 &  SL0        ;

/* State Machine Support Logic                                    */
/* BLAST# can be generated by a condition where the FIFO cannot   */
/* provide or accept data, the fourth data phase of any burst, or */

nBLAST = !(((MASTER3 # MASTER4 # MASTER5) & STOP) # MASTER6);

/* Assert FIFO strobe for entire data phase, except first clock,  */
/* unless the FIFO cannot support the transaction.          */

nRDFIFO = !(HLDA & nREADY & RD & !RDEMPTY &
            (MASTER3 # MASTER4 # MASTER5 # MASTER6));

nWRFIFO = !(HLDA & nREADY & WR & !WRFULL &
            (MASTER3 # MASTER4 # MASTER5 # MASTER6));

/* Write/Read strobe for add-on logic */

W_nR = RWCONT;

/* Drive address in master state 2 */

nADS = !MASTER2;

/* Drive RDY in slave state one */

nRDY_OUT = !(SLAVE1 # SLAVE2);

/* Assert HOLD to the CPU when a DMA request is received from the */
/* S5933 FIFO.                                                    */

HOLD = !MASTER0 & nRESET;
```

```
/*************************************************************/
/* State machine for slave accesses to program ADDR and TXCNT   */
/*************************************************************/

SEQUENCE slave {

/* Idle State */
present S0
      if !nADS.IO & !A21.IO & A20.IO & !HLDA next S1;
      if !nADS.IO &  A21.IO & A20.IO & !HLDA next S2;
      default next S0;

/* Address Access Data Phase */
present S1
      next S3;

/* Transfer Count Access Data Phase */
present S2
      next S3;

/* Recovery Phase */
present S3
      next S0;
}


/*************************************************************/
/* State machine for master access to perform DMA transfers     */
/*************************************************************/

SEQUENCE master {

/* Idle State If a DMA request occurs and TC is not zero, begin   */
/* DMA process.                                                   */
present M0
      if ((RD & !RDEMPTY) # (WR & !WRFULL)) & !DONE next M1;

/* Assert CPU HOLD, wait for acknowledge  */
present M1
      if HLDA next M2;
      if !HLDA next M1;

/* Address Phase */
present M2
      if ONELEFT next M6;     /* 1 Data phase left */
      if TWOLEFT next M5;     /* 2 Data phases left */
      if THREELEFT next M4;   /* 3 Data phases left */
      default next M3;        /* Count is 4 or more */

/* Data Phase 1 */
present M3
      if nRDY_IN next M3;
      if !nRDY_IN next M4;
```

```
/* Data Phase 2 */
present M4
      if nRDY_IN next M4;
      if !STOP & !nRDY_IN next M5;
      if STOP & !nRDY_IN next M7;

/* Data Phase 3 */
present M5
      if nRDY_IN next M5;
      if !STOP & !nRDY_IN next M6;
      if STOP & !nRDY_IN next M7;

/* Data Phase 4 */
present M6
      if !STOP & !nRDY_IN next M7;
      if nRDY_IN next M6;
      if STOP & !nRDY_IN next M7;

/* Recovery Phase */
present M7
      if (nRDY_IN & RD & !RDEMPTY & !DONE) next M2;
      if (nRDY_IN & WR & !WRFULL & !DONE) next M2;
      if (DONE # STOP) & nRDY_IN next M0;
      if !nRDY_IN next M7;
}
```

PLD SIMULATIONS

The following simulation is the output file from the CSIM logic simulator, simulating the PLD code in Appendix A.

```
CSIM(TD): CUPL Simulation Program
Version 4.5a Serial# ED-32930421
Copyright (c) 1983, 1994 Logical Devices, Inc.
CREATED Wed May 24 14:19:48 1995

LISTING FOR SIMULATION FILE: dma_con4.si

   1: Name       DMA;
   2: Partno     ;
   3: Date       5/19/95;
   4: Revision   0;
   5: Designer   JMW;
   6: Company    AMCC;
   7: Assembly   ;
   8: Location   ;
   9: Device     ;
  10:
  11: /*****************************************************************/
  12: /*                                                               */
  13: /*****************************************************************/
  14: /** Allowable Target Device Types :                          */
  15: /*****************************************************************/
  16:
  17: FIELD ADDR = [A31..0];
  18:
  19: FIELD ADDR_COUNTA = [A7,A6,A5,A4,A3,A2];
  20: FIELD ADDR_COUNTB = [A15,A14,A13,A12,A11,A10,A9,A8];
  21: FIELD ADDR_COUNTC = [A21,A20,A19,A18,A17,A16];
  22: FIELD XFER_COUNTA = [C7,C6,C5,C4,C3,C2];
  23: FIELD XFER_COUNTB = [C17,C16,C15,C14,C13,C12,C11,C10,C9,C8];
  24: FIELD master = [MA2,MA1,MA0];
  25: FIELD slave = [SL1,SL0];
                                                26:                        FIELD
ADDR=[A31,A30,A29,A28,A27,A26,A25,A24,A23,A22,A21,A20,A19,A18,A17,A16,A15,A14,
      A13,A12,A11,A10,A9,A8,A7,A6,A5,A4,A3,A2,A1,A0];
  27:
  28: ORDER:
  29:
  30: nRESET,%1,BPCLK,%1,RDEMPTY,WRFULL,%1,HLDA,%1,nBLAST,%1,nRDY_IN,%1,nADS,%3,
      nWRFIFO,nRDFIFO,%1,ADDR,%1,HOLD,%1,W_nR,%1,MA2,MA1,MA0,%1,SL1,SL0,%1,DONE,%1,
      RWCONT,%1,LOADA,LOADC,%1,CNT;
  31:
```

```
=================================================================================
          R      n     nn
      n   DW   n R     WR                                                 R
      R B ER   B D     RD                                                 W LL
      E P MF H L Y n   FF                                  H W        D C OO
      S C PU L A _ A   II                                  O _ MMM SS O O AA C
      E L TL D S I D   FF                                  L n AAA LL N N DD N
      T K YL A T N S   OO              ADDR                D R 210 10 E T AC T
=================================================================================
RESET DEVICE
0001: 0 C 11 0 Z 1 1   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L Z LLL LL H L LL L
0002: 0 C 11 0 Z 1 1   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L Z LLL LL H L LL L
0003: 1 C 11 0 Z 1 1   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L X LLL LL H L LL L
0004: 1 C 11 0 Z 1 1   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L X LLL LL H L LL L
WRITE DMA ADDRESS = 8080
0005: 1 C 11 0 Z 1 0   HH ZZZZZZZ0Z00100000000000000000ZZ L 1 LLL LH H L HL L
0006: 1 C 11 0 Z 1 1   HH ZZZZZZZ0Z00000001000000010000ZZ L 1 LLL HH H L LL L
0007: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL H L LL L
0008: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL H L LL L
WRITE DMA TRANSFER COUNT (READ FROM S5933)
0009: 1 C 11 0 Z 1 0   HH ZZZZZZZ0Z01100000000000000000ZZ L 1 LLL HL H L LH L
0010: 1 C 11 0 Z 1 1   HH ZZZZZZZ1Z00000000000000001111ZZ L 1 LLL HH L H LL L
0011: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL L H LL L
0012: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL L H LL L
0013: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL L H LL L
0014: 1 C 01 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX H 1 LLH LL L H LL L
0015: 1 C 01 0 Z 1 1   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H 1 LLH LL L H LL L
1ST BURST OF ACCESS:  ADDRESS = 8080, TC = 15
0016: 1 C 01 1 H 1 L   HH LLLLLLLLLLLLLLLLHLLLLLLLHLLLLLLL H H LHH LL L H LL L
0017: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0018: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0019: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0020: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL H
```

```
=================================================================================
          R       n     nn
      n   DW    n R     WR                                                  R
      R B ER    B D     RD                                                  W LL
      E P MF H L Y n    FF                                H W         D C OO
      S C PU L A _ A    II                                O _ MMM SS O O AA C
      E L TL D S I D    FF                                L n AAA LL N N DD N
      T K YL A T N S    OO              ADDR              D R 210 10 E T AC T
=================================================================================
0021: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL L
0022: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHH LL L H LL H
0023: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHH LL L H LL L
0024: 1 C 01 1 L 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLH LL L H LL H
0025: 1 C 01 1 L 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLH LL L H LL L
0026: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLL LL L H LL L
0027: 1 C 01 1 H 1 L   HH LLLLLLLLLLLLLLLLHLLLLLLLHLLHLLLL H H LHH LL L H LL L
2ND BURST OF ACCESS:  ADDRESS = 8084, TC = 11
0028: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0029: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0030: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0031: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0032: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL H
0033: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL L
0034: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHH LL L H LL H
0035: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHH LL L H LL L
0036: 1 C 01 1 L 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLH LL L H LL H
0037: 1 C 01 1 L 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLH LL L H LL L
0038: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLL LL L H LL L
0039: 1 C 01 1 H 1 L   HH LLLLLLLLLLLLLLLLHLLLLLLLHLHLLLL H H LHH LL L H LL L
3RD BURST OF ACCESS:  ADDRESS = 8088, TC = 7
0040: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0041: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0042: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0043: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0044: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL H
0045: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL L
0046: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHH LL L H LL H
0047: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHH LL L H LL L
0048: 1 C 01 1 L 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLH LL L H LL H
0049: 1 C 01 1 L 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLH LL L H LL L
0050: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLL LL L H LL L
0051: 1 C 01 1 H 1 L   HH LLLLLLLLLLLLLLLLHLLLLLLLHLHHLLLL H H LHH LL L H LL L
4TH BURST OF ACCESS:  ADDRESS = 808C, TC = 3
0052: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL L
0053: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL L
0054: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL L
0055: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL L
0056: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHH LL L H LL H
0057: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHH LL L H LL L
0058: 1 C 01 1 L 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLH LL L H LL H
0059: 1 C 01 1 L 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLH LL L H LL L
0060: 1 C 01 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLL LL H H LL L
```

```
0061: 1 C 01 1 H 1 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L H LLL LL H H LL L
0062: 1 C 01 1 H 1 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L H LLL LL H H LL L
0063: 1 C 01 1 H 1 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L H LLL LL H H LL L
WRITE DMA ADDRESS
0064: 1 C 11 0 Z 1 0   HH ZZZZZZZ0Z00100000000000000000ZZ L 1 LLL LH H H HL L
0065: 1 C 11 0 Z 1 1   HH ZZZZZZZ0Z00000000101010101010ZZ L 1 LLL HH H H LL L
0066: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL H H LL L
WRITE DMA TRANSFER COUNT (WRITE TO S5933)
0067: 1 C 11 0 Z 1 0   HH ZZZZZZZ0Z01100000000000000000ZZ L 1 LLL HL H H LH L
0068: 1 C 11 0 Z 1 1   HH ZZZZZZZ0Z00000000000000000101ZZ L 1 LLL HH L L LL L
0069: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL L L LL L
0070: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL L L LL L
0071: 1 C 10 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX H 1 LLH LL L L LL L
0072: 1 C 10 0 Z 1 1   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H 1 LLH LL L L LL L
1ST BURST OF ACCESS:  ADDRESS = 5550, TC = 5
0073: 1 C 10 1 H 1 L   HH LLLLLLLLLLLLLLLLLHLHLHLHLHLHLLLL H L LHH LL L L LL L
0074: 1 C 10 1 H 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L LHL LL L L LL L
0075: 1 C 10 1 H 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L LHL LL L L LL L
0076: 1 C 10 1 H 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L LHL LL L L LL L
0077: 1 C 10 1 H 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L LHL LL L L LL L
0078: 1 C 10 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HHL LL L L LL H
0079: 1 C 10 1 H 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HHL LL L L LL L
0080: 1 C 10 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HHH LL L L LL H
0081: 1 C 10 1 H 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HHH LL L L LL L
0082: 1 C 10 1 L 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HLH LL L L LL H
0083: 1 C 10 1 L 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HLH LL L L LL L
0084: 1 C 10 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HLL LL L L LL L
0085: 1 C 10 1 H 1 L   HH LLLLLLLLLLLLLLLLLHLHLHLHLHHLLLLL H L LHH LL L L LL L
2ND BURST OF ACCESS:  ADDRESS = 5554, TC = 1
```

```
=================================================================================
          R      n     nn
      n   DW   n R     WR                                                  R
      R B ER   B D     RD                                                  W LL
      E P MF H L Y n   FF                                  H W        D C OO
      S C PU L A _ A   II                                  O _ MMM SS O O AA C
      E L TL D S I D   FF                                  L n AAA LL N N DD N
      T K YL A T N S   OO               ADDR               D R 210 10 E T AC T
=================================================================================
0086: 1 C 10 1 L 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HLH LL L L LL L
0087: 1 C 10 1 L 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HLH LL L L LL L
0088: 1 C 10 1 L 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HLH LL L L LL L
0089: 1 C 10 1 L 1 H   LH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HLH LL L L LL L
0090: 1 C 10 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H L HLL LL H L LL L
0091: 1 C 10 1 H 1 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L L LLL LL H L LL L

RDEMPTY ASSERTED DURING A READ TRANSFER

WRITE DMA ADDRESS = 8080
0092: 1 C 11 0 Z 1 0   HH ZZZZZZZ0Z00100000000000000000ZZ L 1 LLL LH H L HL L
0093: 1 C 11 0 Z 1 1   HH ZZZZZZZ0Z00000001000000010000ZZ L 1 LLL HH H L LL L
0094: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL H L LL L
0095: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL H L LL L
WRITE DMA TRANSFER COUNT (READ FROM S5933)
0096: 1 C 11 0 Z 1 0   HH ZZZZZZZ0Z01100000000000000000ZZ L 1 LLL HL H L LH L
0097: 1 C 11 0 Z 1 1   HH ZZZZZZZ1Z00000000000000001111ZZ L 1 LLL HH L H LL L
0098: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL L H LL L
0099: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL L H LL L
0100: 1 C 11 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX L 1 LLL LL L H LL L
0101: 1 C 01 0 Z 1 1   HH XXXXXXXXXXXXXXXXXXXXXXXXXXXXXX H 1 LLH LL L H LL L
0102: 1 C 01 0 Z 1 1   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H 1 LLH LL L H LL L
1ST BURST OF ACCESS:  ADDRESS = 8080, TC = 15
0103: 1 C 01 1 H 1 L   HH LLLLLLLLLLLLLLLLHLLLLLLLHLLLLLL H H LHH LL L H LL L
0104: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0105: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0106: 1 C 01 1 H 1 H   HL ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H LHL LL L H LL L
0107: 1 C 11 1 L 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL L
0108: 1 C 11 1 L 1 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HHL LL L H LL L
0109: 1 C 11 1 H 0 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ H H HLL LL L H LL L
0110: 1 C 11 1 H 1 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L H LLL LL L H LL L
0111: 1 C 11 1 H 1 H   HH ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ L H LLL LL L H LL L
```

Careful attention must always be exercised when beginning the design and layout phase of any new printed circuit board. It is only through careful planning and the usage of good electrical design practices that long term product reliability may be achieved. Careful attention to layout issues of EMI, cross-talk and decoupling will avoid engineering prototype development problems and future production failures due to an "on the edge" design.

Designs incorporating PCI devices have specific printed circuit board layout requirements in order to comply with industry standard PCI local bus specifications. In addition to these requirements, AMCC's S5933 PCI controller has specific requirements to ensure proper function and long life. This applications note is supplied by AMCC as factory recommended PCB design layout guidelines for printed circuit boards incorporating the S5933 PCI controller. The following sections detail important design areas concerning PCB design, power decoupling, ferrite beads and critical trace layout.

PCB DESIGN

AMCC highly recommends the use of a four layer printed circuit board. This is due to the expected high trace density in most PCI designs incorporating PLCC and PQFP devices. Four layer designs significantly overcome ground noise problems associated with most two layer PCBs. Should a two layer design be implemented, leave as much copper on the PCB as possible for all power distribution traces. This is extremely important in ground traces to avoid ground loops and ground potential problems. Also utilize multiple feed-thrus for large traces. A 100 mil trace with a single 0.030 feed thru has it's current carrying capability significantly reduced.

DECOUPLING

The AMCC PCI controller is an application specific standard product (ASSP) utilizing CMOS technology. Although CMOS technology is commonly known for it's noise immunity properties, special consideration must still be given to electrical noise generated due to the high speed signals utilized in a PCI design.

The first design issue of concern is decoupling. By "decoupling", the designer provides a means to dissociate circuit functions from the power bus serving that circuit. This "means" is provided through the usage of decoupling capacitors, ferrite beads and proper printed circuit board trace layout. The lack of correct decoupling increases both radiated and conducted emissions which increases electrical noise and susceptibility to circuit failure (i.e. erratic and intermittent circuit functions or "FLAKYNESS").

To reduce these problems, AMCC recommends PCI designs incorporate a low speed PCB decoupling capacitor. Select a 10uF (minimum) tantalum or metalized polycarbonate (not aluminum) capacitor for this purpose. Specify a low ESR type at a working voltage slightly above the circuit's operating voltage. Locate the capacitor no more than 300 mils from the PCB's power entry point as shown in figure 1. This point is likely the PCB edge fingers or a wire connector interfacing the PCB to the system power supply.

AMCC also recommends the use of one high speed 0.1uF decoupling capacitor per PCB IC package. Specify an X7R or BX type dielectric ceramic chip capacitor also rated slightly above required working voltage for this purpose. Locate each capacitor next to and on the same side of the PCB as each IC device. Locate capacitors no more than 150 mils from each IC package's ground pin. High speed decoupling for the S5933 PCI controller IC requires one 0.1uF per power and ground pin pair. IMPORTANT: Locate these capacitors on the same side of the PCB as the S5933 at a distance of less than 150 mils as shown in Figure 1.

The use of the above capacitors will sufficiently decouple the Vcc and ground planes from high and low speed circuit functions. One last decoupling issue of concern involves any unused PCI edge connector power fingers. Industry PCI specifications provides for power sources of +3.3V and +5V within the PCI edge connector. PCI specifications require any unused power and V I/O pins on the PCI edge connector be decoupled to the ground plane with an average of 0.01uF.

FERRITE BEADS

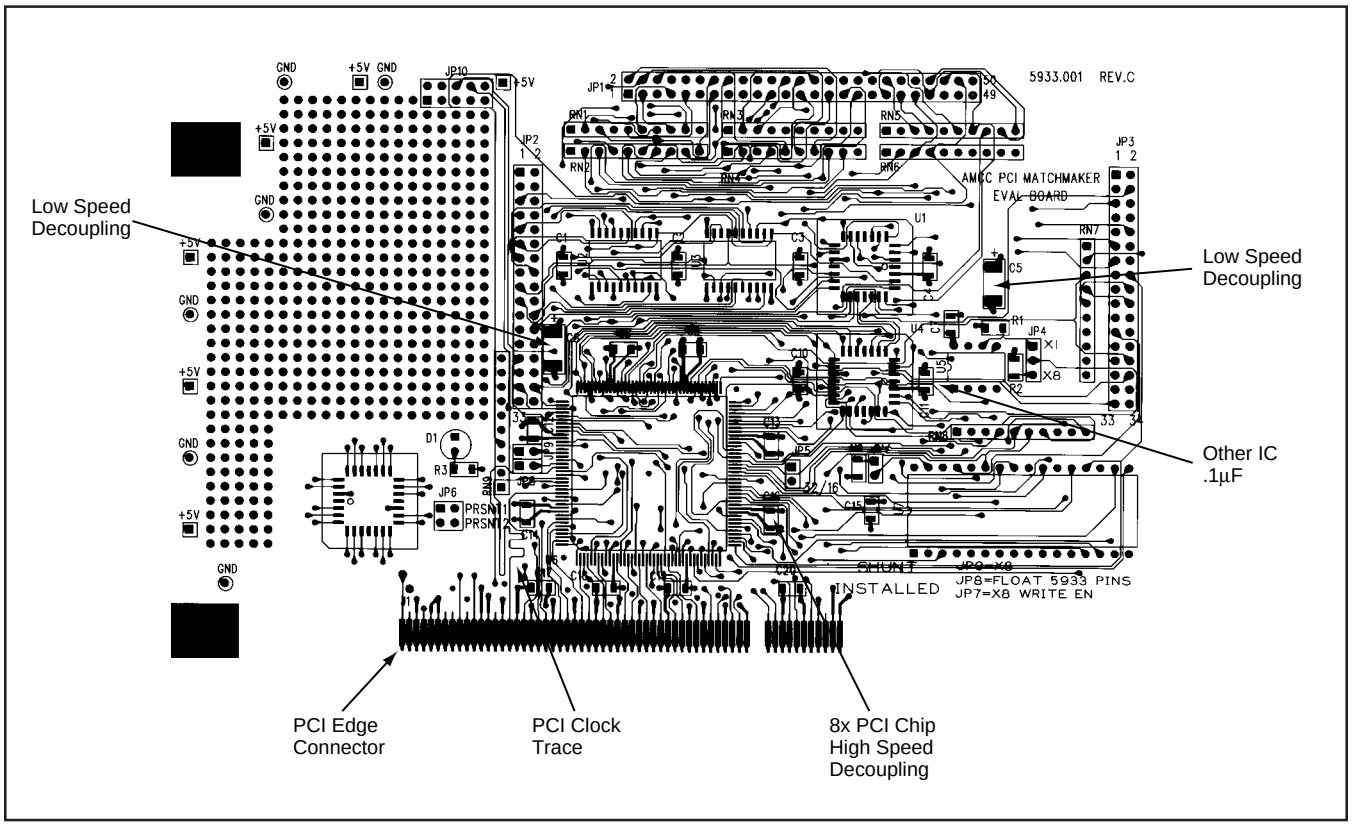
In many applications the requirement for ferrite beads to reduce high frequency noise is unnecessary once the above indicated board layout and bypass practices have been followed. However, in some applications high frequency noise may persist. In these cases the addition of a ferrite bead to the input +V power entry point as shown in figure 1 may be necessary. The size and type of material used will vary depending on the particular design and electrical noise to be eliminated. Consult the bead manufacturer data books to determine the best fit. Expect to try a small selection of beads to achieve best results. Use a single pass thru or one turn to increase effectiveness of the bead. **DO NOT** series beads to increase impedance. Use a cracked air gap bead to reduce saturation effects as necessary.

CRITICAL TRACE LAYOUT

AMCC's S5933 PCI controller is very powerful and flexible. It operates at clock and data bus speeds of up to 33 megahertz. Data bus transfer operations can occur in 30 nanoseconds with signal rise and fall times of 3 nanoseconds. At these speeds, signal traces look more like transmission lines where trace impedance becomes an important factor. Carefull attention to trace lengths and routing will prevent impedance caused ringing and will preserve signal rise and fall times. As a last design issue, standard PCI specifications require the following design criteria be adhered to for all PCI bus controller devices.

1. Trace lengths for each 32 bit interface data signal from the S5933 to the PCI bus is limited to a maximum of 1.5 inches for all 32 bit and 64 bit cards.
2. Trace lengths for the balance of the PCI bus signals, used in the 64 bit extension, is limited to a maximum of 2.0 inches from the S5933 to the PCI bus.
3. The trace connecting the S5933 PCI clock signal (S5933 pin 142) to the PCI bus connector must be 2.5 inches +/- .1 inches in length and can be routed to only one load. It may be necessary to "snake" this trace, as shown in figure 1, depending on the physical location of the PCI controller IC on the PCB to ensure this requirement is met. Ensure all corners of this trace are rounded. Do not use 90 degree sharp corners.
4. The unloaded impedance of a shared PCI signal trace on the expansion card must be held within a 60 to 100 ohm range.

Figure 1. PCI Evaluation Board Signal Layer #1



CONTENTS

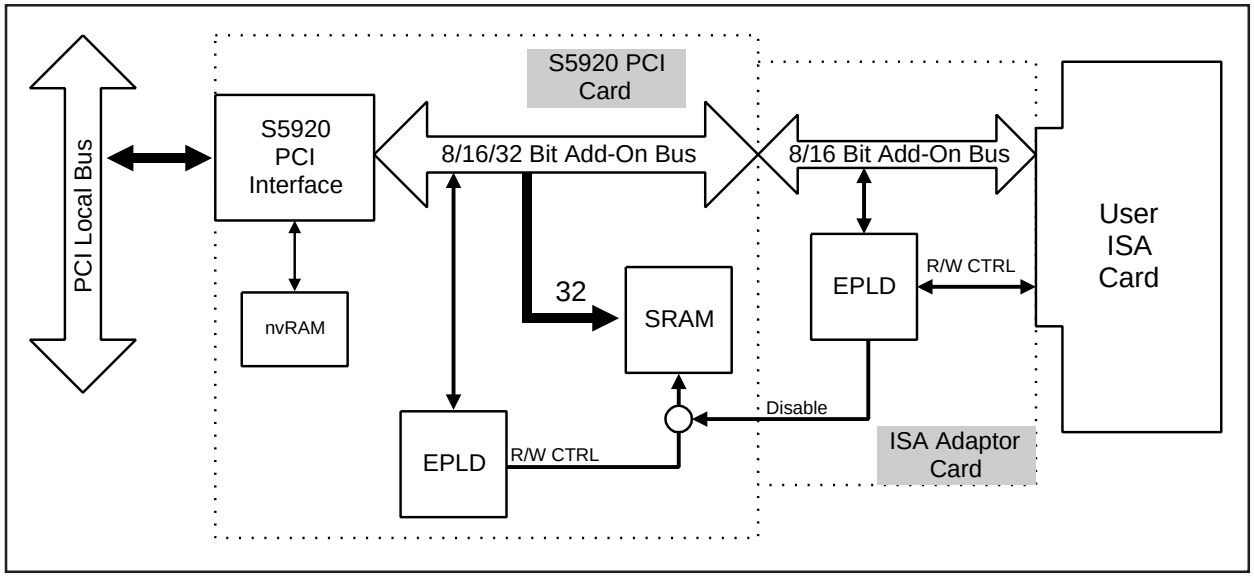
Introduction

The AMCC PCI Developer's Kit contains everything needed for the *PCI* developer to immediately begin operating and experimenting with a S5920 based *PCI* design. For software engineers, the Developer's Kit is a fully functional PCI to Add-On bus test card. The programmer can immediately begin testing and operating numerous aspects of PCI Bus to Add-On Bus data transfers, timings, control and overall operation. The programmer can also test and become familiar with the various aspects of PCI BIOS functions and PCI Configuration Space operation. A set of DOS based development programs allow the programmer to view and change device register contents from the PCI Bus as well as view and change PCI configurations. Additional development software provides downloading, editing, configuring and programming capability to the optional serial boot load nvRAM contained on the main Developer Kit PCI card.

For the hardware designer, the Developer's Kit provides fully functional PCI to Add-On bus design examples. The main S5920 PCI card shows Add-On bus connection to onboard SRAM. The ISA Adapter card allows the designer to plug in an existent ISA card to the Add-On bus to begin logic optimization and reduction.

The Developer's Kit comes complete with schematics, PCB artwork, EPLD equation source code for each application example and development software source code. The designer is able to implement portions or all of an Add-On bus design using a supplied bread board. Extra headers and EPLD sockets are available in the Developer Kit to further assist in proto-typing and general experimentation. Dedicated Hewlett Packard PCI logic analyzer headers are provided for directly cable connection.

Developer's Kit Block Diagram



Developer's Kit Overview

The PCI Developer's kit contains two printed circuit boards plus a software tools diskette. The S5920 PCI card contains an S5920, SRAM and a pre-programmed EPLD containing Add-On bus control functions. This card was developed to demonstrate interconnection of the S5920 PCI interface chip to the PCI Bus and interconnection of the S5920's Add-On Bus to a basic SRAM design. The onboard EPLD is specifically programmed to control the Add-On bus for Active Mode data transfers for burst or single cycle data reads and writes to the SRAM. The Add-On bus signals are also routed to a set of four external application connectors. These connectors provide the designer with additional Add-On bus connection capability. The designer can utilize these for attaching his/her own application PCB to the PCI card's Add-On bus. Two of these connectors are designed to provide simultaneous connection of the user's PCB and a logic analyzer. Although many logic analyzers may be connected, the are designed specifically for connection directly with Hewlett Packards PCI logic analyzer pod cabling.

The second PCB is an ISA Adapter interface card designed specifically to mate with the S5920 PCI card. The adapter card was developed to provide direct connect of many existent ISA cards to the S5920 Add-On bus. An adapter card EPLD is programmed to convert Add-On bus signals to ISA card signals and vise versa. The adapter card provides the designer with a basic functioning interface example to the PCI bus allowing the designer to start design optimization and logic reduction. The programmer can immediately begin reading and writing data from the PCI bus to ISA card addresses.

It is important for the designer to remember, the developer's kit was designed to demonstrate various aspects of S5920 user design. The specific EPLDs, Add-On logic components and software was chosen to support multiple application illustrations. Therefore, the device costs and complexity is more than will be necessary for many applications.

PCI MATCHMAKER CONTROLLER DEVELOPER'S KIT S5933DK1

Developer's Kit Includes:

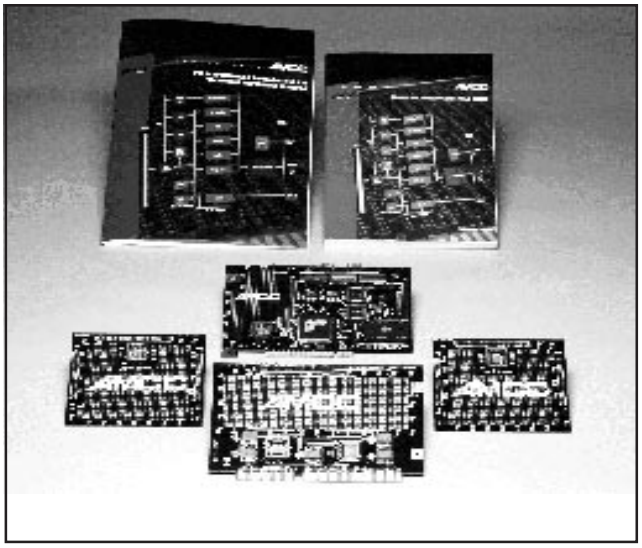
- PCI Revision 2.1 compliant circuit board with an S5933 controller, two 22V10 PLDs, one byte-wide FLASH memory and serial nvRAM
- Two breadboard "daughter cards" with connectors
- One ISA "loopback" card with ribbon cable
- Developer's Kit Technical Reference Manual including PLD equations and board schematics
- S5933 PCI Controller data book

General Description

The S5933DK1 PCI Controller Developer's Kit provides a proven reference platform for use in prototyping applications. The kit has a modular design that allows hardware "breadboarding" as well as general evaluation of the S5933 PCI Matchmaker Controller. Diagnostic and development utility software is included to help reduce time-to-market for both new and existing applications.

The S5933 demonstration board consists of a Revision 2.1 PCI Specification compliant edge connector and an S5933 device packaged in a 160-pin plastic quad flat pack (PQFP). In addition, two preprogrammed 22V10 Programmable Logic Devices (PLD's) are provided to generate user bus timing signals. Both preprogrammed serial and byte-wide nvRAMs are included and may be reprogrammed by the user.

Via on-board connectors, the S5933 board can be interfaced to either of two breadboard daughter cards that are included in the kit. Each daughter card has a PLD socket along with 3 x 4 inch wire wrap area. Alternatively, the evaluation board can be interfaced with the ISA loopback card provided. The ISA card consists of a standard edge connector and a ribbon cable which connects to the S5933 evaluation board.



The ISA card allows the user to "emulate" an Add-On processor using either the host system or a second PC. The loopback feature provides direct access to the S5933's Add-On bus, enabling the user to pass commands and data through the S5933 to the PCI bus. The breadboard area provided by the ISA card allows for customized functionality. Software developers' tools included with the kit allow the user to exercise various configurations and controller registers using a DOS interface, AMCCDIAG, a diagnostic program which also verifies the integrity of the S5933 data. The NV Build program assists in the in-system programming of the external nonvolatile memory to implement application-specific configurations.

A user's manual and full design documentation complete the kit.

SECTION 6: Sales Information

CONTENTS

AMCC SALES OFFICES

Europe/Israel

101 Wright Road
Hollis, NH 03049
Tel: 603/465-6138
Fax: 603/465-3650

ASIA Pacific & ROW

Applied Micro Circuits Corp.
Corporate Headquarters
6290 Sequence Drive
San Diego, CA 92121
Tel: 619/450-9333
Fax: 619/450-9885

Northeast

Applied Micro Circuits Corp.
25 Burlington Mall Road
Suite 300
Burlington, MA 01803
Tel: 781/270-0674
Fax: 781/221-5853

Mid-America, Arizona & New Mexico

Applied Micro Circuits Corp.
840 E. Central Parkway
Suite 120
Plano, TX 75074
Tel: 972/423-7989
Fax: 972/424-6617

Northwest

Applied Micro Circuits Corp.
950 S. Bascom Avenue
Suite 1113
San Jose, CA 95128
Tel: 408/289-1190
Fax: 408/289-1527

Southwest

Applied Micro Circuits Corp.
501 N. El Camino Real
Suite 217
San Clemente, CA 92672
Tel: 714/366-4105
Fax: 714/366-4126

Southeast

Applied Micro Circuits Corp.
79 Alexander Drive
Bldg. 4401, Suite 200
P.O. Box 14088
Research Triangle Park, NC 27709
Tel: 919/558-2003
Fax: 919/558-2004

NORTH AMERICAN SALES REPRESENTATIVES

ALABAMA

Glen White Associates
3322 S. Memorial Pkwy.
Suite 67
Huntsville, AL 35801
Tel: 205/882-6751
Fax: 205/880-2750

ARIZONA

Quatra Associates, Inc.
10235 S. 51st Street
Suite 160
Phoenix, AZ 85044
Tel: 602/753-5544
Fax: 602/753-0640

CALIFORNIA

Harper & Two
9845 Erma Rd
Suite 311
San Diego, CA 92131
Tel: 619/549-5366
Fax: 619/549-5365

Harper & Two
2798 Junipero Avenue
Signal Hill, CA 90806
Tel: 562/424-3030
Fax: 562/424-6622

Centaur North
2890 Zanker Road, Suite 203
San Jose, CA 95134
Tel: 408/894-0182
Fax: 408/894-0178

Sierratek Marketing
11531 Sun Valley Road
Truckee, CA 96161
Tel: 916/587-8360
Fax: 916/587-8361

CANADA

Electronic Sales Professionals, Inc. (ESP)
215 Stafford Road West
Unit 104
Nepean, Ontario
Canada K2H 9C1
Tel: 613/828-6881
Fax: 613/828-5725

Electronic Sales Professionals, Inc. (ESP)
4210 Sere
Suite 210
St. Laurent, Quebec
Canada H4T 1A6
Tel: 514/344-0420
Fax: 514/344-4914

(British Columbia)
L2 (L-Squared LTD)
11643 A NE 70th Place
Kirkland, WA 98033
Tel: 206/525-8555
Fax: 206/527-0882

COLORADO

Luscombe Engineering Co.
1500 Kansas Ave., Suite 1B
Longmont, CO 80501
Tel: 303/772-3342
Fax: 303/772-8783

Luscombe Engineering Co.
2376 Trails End
Franktown, CO 80116
Tel: 303/814-9725
Fax: 303/814-9456

CONNECTICUT

Dynamic Technologies
33 North Cove Road
Old Saybrook, CT 06475
Tel: 860/388-0130
Fax: 860/388-9406

DELAWARE

Delta Technical Sales, Inc.
122 North York Road
Suite 9
Hatboro, PA 19040
Tel: 215/957-0600
Fax: 215/957-0920

FLORIDA

Mega Technologies, Inc.
1600 Sarno Road
Suite 21
Melbourne, FL 32935
Tel: 407/752-6767
Fax: 407/752-7484

Mega Technologies, Inc.
815 NE 1st Court
Delray Beach, FL 33483
Tel: 561/278-6513
Fax: 561/278-6549

Mega Technologies, Inc.
2510 Cypress Bend Drive
Clearwater, FL 34621
Tel: 813/797-8222
Fax: 813/797-8315

GEORGIA

Glen White Associates
2444 Highway 120
Suite 101
Duluth, GA 30155
Tel: 770/418-1500
Fax: 770/418-1660

IDAHO

L2 (L-Squared LTD)
15247 NW Greenbrier Pkwy.
Beaverton, OR 97006
Tel: 503/629-8555
Fax: 503/645-6196

ILLINOIS

Phase II Marketing, Inc.
2260 Hicks Road
Suite 410
Rolling Meadows, IL 60008
Tel: 847/577-9401
Fax: 847/577-9491

Customer 1st, Inc.
10540 Marty
Suite 200N
Overland Park, KS 66212
Tel: 913/895-9593
Fax: 612/851-7907

INDIANA

Century Technical Sales, Inc.
3520 W. 86th Street
Suite 260
Indianapolis, IN 46268
Tel: 317/876-0101
Fax: 317/875-5566

IOWA

Customer 1st
2950 Metro Drive
Suite 101
Bloomington, MN 55425
Tel: 612/851-7909
Fax: 612/851-7907

KANSAS

Customer 1st, Inc.
10540 Marty
Suite 200N
Overland Park, KS 66212
Tel: 913/895-9593
Fax: 612/851-7907

KENTUCKY

Century Technical Sales, Inc.
845 Lane Allen Road
Suite 13
Lexington, KY 40504
Tel: 606/276-3164
Fax: 606/276-4033

ERA, Inc.
354 Veterans Memorial Hwy.
Commack, NY 11725
Tel: 516/543-0510
Fax: 516/543-0758

MAINE

Comp Rep Associates
100 Everett Street
Westwood, MA 02090
Tel: 781/329-3454
Fax: 781/329-6395

MARYLAND

Cetan
22 West Padonia Road
Suite B 317
Timonium, MD 21093
Tel: 410/453-0969
Fax: 410/453-6199

MASSACHUSETTS

Comp Rep Associates
100 Everett Street
Westwood, MA 02090
Tel: 781/329-3454
Fax: 781/329-6395

MICHIGAN

Century Technical Sales, Inc.
43422 West Oaks Drive
Suite 270
Novi, MI 48377
Tel: 248/344-2550
Fax: 248/344-2815

MINNESOTA

Customer 1st
2950 Metro Drive
Suite 101
Bloomington, MN 55425
Tel: 612/851-7909
Fax: 612/851-7907

MISSISSIPPI

Glen White Associates
3322 S. Memorial Pkwy.
Suite 67
Huntsville, AL 35801
Tel: 205/882-6751
Fax: 205/880-2750

MISSOURI

Customer 1st, Inc.
10540 Marty
Suite 200N
Overland Park, KS 66212
Tel: 913/895-9593
Fax: 612/851-7907

MONTANA

L2 (L-Squared LTD)
15247 NW Greenbrier Pkwy.
Beaverton, OR 97006
Tel: 503/629-8555
Fax: 503/645-6196

NEBRASKA

Customer 1st, Inc.
10540 Marty
Suite 200N
Overland Park, KS 66212
Tel: 913/895-9593
Fax: 612/851-7907

NEVADA

Quatra Associates, Inc.
10235 S. 51st Street
Suite 160
Phoenix, AZ 85044
Tel: 602/753-5544
Fax: 602/753-0640

Centaur North
2890 Zanker Road, Suite 203
San Jose, CA 95134
Tel: 408/894-0182
Fax: 408/894-0178

NEW HAMPSHIRE

Comp Rep Associates
100 Everett Street
Westwood, MA 02090
Tel: 781/329-3454
Fax: 781/329-6395

NEW JERSEY

(Southern New Jersey)
Delta Technical Sales, Inc.
122 North York Road
Suite 9
Hatboro, PA 19040
Tel: 215/957-0600
Fax: 215/957-0920

(Northern New Jersey)
ERA, Inc.
354 Veterans Memorial Hwy.
Commack, NY 11725
Tel: 516/543-0510
Fax: 516/543-0758

NEW MEXICO

Quatra Associates, Inc.
600 Autumnwood Place S.E.
Albuquerque, NM 87123-4347
Tel: 505/296-6781
Fax: 505/292-2092

NEW YORK

Quality Components
116 Fayette Street
Manlius, NY 13104
Tel: 315/682-8885
Fax: 315/682-2277

ERA, Inc.
354 Veterans Memorial Hwy.
Commack, NY 11725
Tel: 516/543-0510
Fax: 516/543-0758

NORTH CAROLINA

Glen White Associates
6070J Six Forks Road
Raleigh, NC 27609
Tel: 919/848-1931
Fax: 919/847-3294

Glen White Associates
13420 West Reese Blvd.
Huntersville, NC 28078
Tel: 704/875-3777
Fax: 704/875-3843

NORTH DAKOTA

Customer 1st
2950 Metro Drive
Suite 110
Bloomington, MN 55425
Tel: 612/851-7909
Fax: 612/851-7907

OHIO

Century Technical Sales, Inc.
8977 Columbia Road
Suite G
Loveland, OH 45140
Tel: 513/677-5088
Fax: 513/677-1775

Century Technical Sales, Inc.
24600 Center Ridge Road
Suite 120
Westlake, OH 44145
Tel: 216/808-9171
Fax: 216/871-9681

Century Technical Sales, Inc.
6161 Busch Blvd. #110
Columbus, OH 43229
Tel: 614/433-7500
Fax: 614/433-9085

OKLAHOMA

Logic 1 Sales, Inc.
101 W. Renner Road
Suite 190
Richardson, TX 75082
Tel: 972/234-0765
Fax: 972/669-3042

OREGON

L2 (L-Squared LTD)
15247 NW Greenbrier Pkwy.
Beaverton, OR 97006
Tel: 503/629-8555
Fax: 503/645-6196

PENNSYLVANIA

Century Technical Sales, Inc.
10592 Perry Hwy, Suite 217
Wexford, PA 15090
Tel: 412/934-2326
Fax: 412/934-3031

(Eastern Pennsylvania)
Delta Technical Sales, Inc.
122 North York Road
Suite 9
Hatboro, PA 19040
Tel: 215/957-0600
Fax: 215/957-0920

RHODE ISLAND

Comp Rep Associates
100 Everett Street
Westwood, MA 02090
Tel: 781/329-3454
Fax: 781/329-6395

SOUTH CAROLINA

Glen White Associates
13420 West Reese Blvd.
Huntersville, NC 28078
Tel: 704/875-3777
Fax: 704/875-3843

SOUTH DAKOTA

Customer 1st
2950 Metro Drive
Suite 101
Bloomington, MN 55425
Tel: 612/851-7909
Fax: 612/851-7907

TENNESSEE

Glen White Associates
3322 S. Memorial Pkwy.
Suite 67
Huntsville, AL 35801
Tel: 205/882-6751
Fax: 205/880-2750

TEXAS

Logic 1 Sales, Inc.
101 W. Renner Road
Suite 190
Richardson, TX 75082
Tel: 972/234-0765
Fax: 972/669-3042

Logic 1 Sales, Inc.
9111 Jollyville Road
Suite 112
Austin, TX 78759-7433
Tel: 512/ 345-2952
Fax: 512/346-5309

Logic 1 Sales, Inc.
4606 FM 1960 West
Suite 308
Houston, TX 77069
Tel: 281/444-7594
Fax: 281/444-8236

UTAH

First Source
8341 S. 700 East
Sandy, UT 84070
Tel: 801/561-1999
Fax: 801/561-4525

VERMONT

Comp Rep Associates
100 Everett Street
Westwood, MA 02090
Tel: 781/329-3454
Fax: 781/329-6395

VIRGINIA

Cetan
22 West Padonia Road
Suite B 317
Timonium, MD 21093
Tel: 410/453-0969
Fax: 410/453-6199

WASHINGTON

L2 (L-Squared LTD)
11643 A NE 70th Place
Kirkland, WA 98033
Tel: 206/525-8555
Fax: 206/527-0882

WASHINGTON, D.C.

Cetan
22 West Padonia Road
Suite B 317
Timonium, MD 21093
Tel: 410/453-0969
Fax: 410/453-6199

WISCONSIN

Phase II Marketing, Inc.
205 Bishops Way
Suite 220
Brookfield, WI 53005
Tel: 414/797-9986
Fax: 414/797-9935

(Western Wisconsin)
Customer 1st
2950 Metro Drive
Suite 110
Bloomington, MN 55425
Tel: 612/851-7909
Fax: 612/851-7907

NORTH AMERICAN DISTRIBUTOR

Insight Electronics
1-800-677-6011 ext. 94
http://www.ikn.com

INTERNATIONAL SALES REPRESENTATIVES

UNITED KINGDOM

AMEGA Electronics, LTD.
Loddon Business Centre
Roentgen Road,
Daneshill East
Basingstoke
Hants RG24 8NG
United Kingdom
Tel: 44-1256-305330
Fax: 44-1256-305335

ITALY

ACSIS s.r.l.
Via Alberto Mario, 26
20149 Milano, Italy
Tel: 39-248022522
Fax: 39-248012289

ESCO Italiana SPA
Viale Fratelli Casiraghi, 355
Sesto San Giovanni
20099 Milano, Italy
Tel: 39-2-2409241
Fax: 39-2-2409255

FRANCE

A2M
Siege Social
5 rue Carle Vernet
92315 Sevres Cedex
France
Tel: 33-1-46-237900
Fax: 33-1-46-237923

Sil Design
14 Avenue du Quebec
Batiment K1
B.P. 724 Villebon
91961 Coutaboeuf Cedex
France
Tel: 33-1-644-63576
Fax: 33-1-690-74545

FINLAND

Yleiselektroniikka Oy
Luomannotko 6
FIN-02201 ESPOO
Finland
Tel: 358-9-4526-21
Fax: 358-9-4526-2202

GERMANY

Tekelec Airtronic GmbH
Kapuzinerstrasse 9
80337 Munich
Germany
Tel: 49-89-51640
Fax: 49-89-535129

NORWAY

Bit Elektronikk A/S
Smedsvingen 4
P.O. Box 194
1360 Nesbru
Norway
Tel: 47-66-77-65-00
Fax: 47-66-77-65-01

SWITZERLAND/ AUSTRIA

Ixlogic AG
Badenerstrasse 808
CH-8048
Zurich, Switzerland
Tel: 41-1 434-78-10
Fax: 41-1 434-78-19

DENMARK

Dan-Contact
Brogaardsvej 48
DK-2820 Gentofte
Denmark
Tel: 45-39-683633
Fax: 45-39-683362

NETHERLANDS/ BELGIUM

Tekelec Airtronic B.V.
Ypsilon House
Engelandlaan 310
2711 DZ Zoetermeer
Nederland
Tel: 31-79-3461430
Fax: 31-79-3417504

SOUTH AFRICA

Insight Electronics Pty. Ltd.
Melbourne
Unit 2, 14 Melrich Road
Bayswater, Victoria 3153
Australia
Tel: 61-3-9761-3455
Fax: 61-3-9761-3415

SWEDEN

DipCom Electronics
Torshamnsgatan 35
Box 1230
S-16428 KISTA
Sweden
Tel: 46-8-7522480
Fax: 46-8-7513649

AUSTRALIA

Insight Electronics Pty. Ltd.
Melbourne
Unit 2, 14 Melrich Road
Bayswater, Victoria 3153
Australia
Tel: 61-3-9761-3455
Fax: 61-3-9761-3415

NEW ZEALAND

Insight Electronics Pty. Ltd.
Auckland
Unit 7, 110 Mays Road
Penrose, Auckland
New Zealand
Tel: 64-9-636-5984
Fax: 64-9-636-5985

ISRAEL

Eldis Technologies
26 Arlozorov Street
POB 200 Herzlia 46101
Israel
Tel: 972-9-9562666
Fax: 972-9-9562642

Accord Technologies, Inc.
777 Old Country Road
Plainview, NY 11803
Tel: 516/933-6646
Fax: 516/933-6645

CHINA

Twin-Star Trading Co., Ltd.
Room B, 26/FI, Lever Centre
69-71 King Yip Street
Hong Kong, S.A.R.
China
Tel: 852-2341-4282
Fax: 852-2763-7717

JAPAN

Teksel Co., LTD.
TBC
Higashi 2-27-10
Shibuya-ku, Tokyo 150
JAPAN
Tel: 81-3-5467-9104
Fax: 81-3-5467-9346

KOREA

Buksung Industrial Co., LTD.
7F Saehan Bldg.
1653-6 Shinlimdong
Kwanak-Ku, Seoul,
Korea
Tel: 82-2-866-1360
Fax: 82-2-862-1273

SINGAPORE/ MALAYSIA/ THAILAND

Gates Engineering Pte, LTD.
1123 Serangoon Road
#03-01 UMW Building
Singapore 328207
Tel: 65-299-9937
Fax: 65-299-7636

TAIWAN

Promate Electronic Co., LTD.
4F, 32, Sec.1 Huan Shan Road
Nei Hu, Taipei 114
Taiwan, R.O.C.
Tel: 886-2-6590303
Fax: 886-2-6580988

INDIA

Interex India
No: 58, II Floor
III Cross II Phase, J.P. Nagar
Bangalore – 560 078
India
Tel: 91-80 640-663
Fax: 91-80 672-1432

Interex India/USA
2629 Terminal Blvd.
Mountain View, CA 94049
Tel: 650/254-0627
Fax: 650/254-1540