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

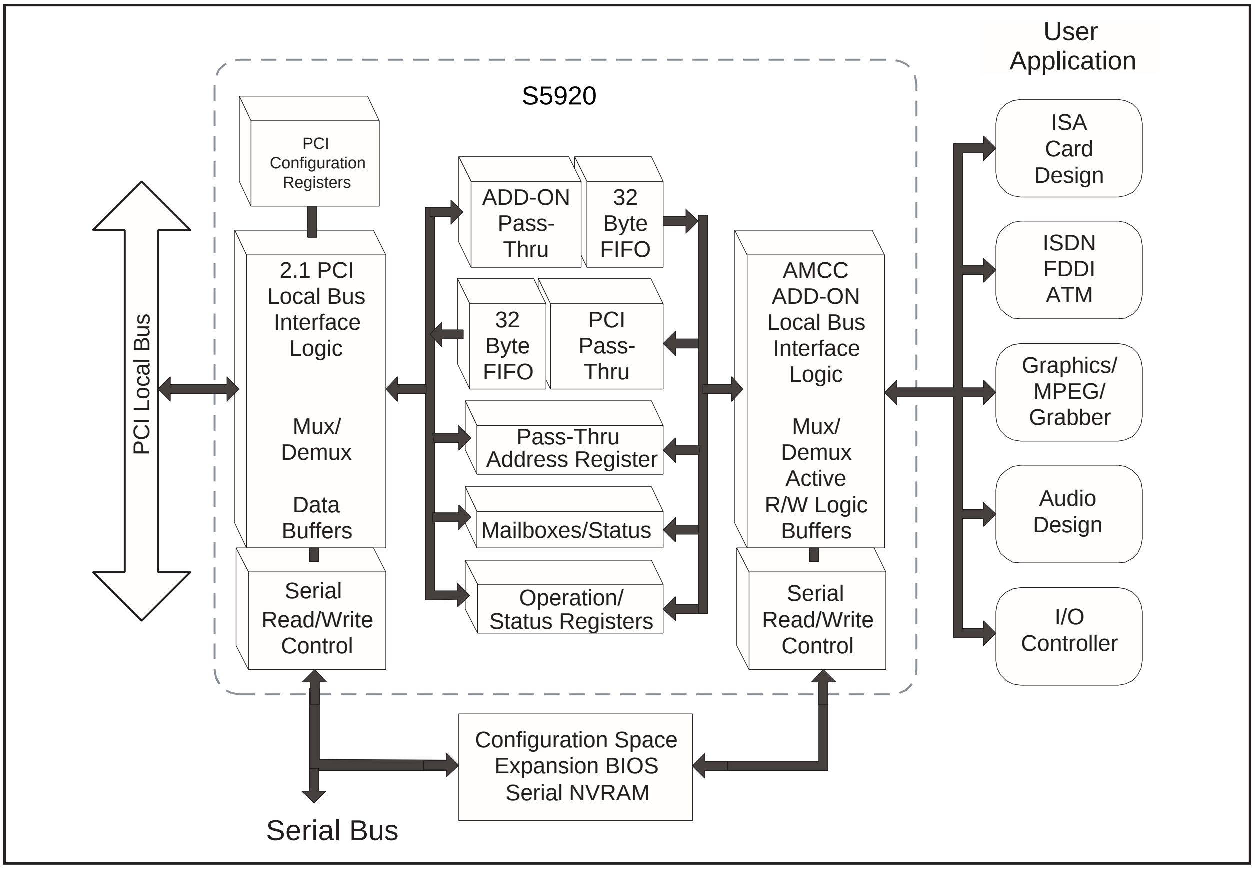
- ISA Conversions
- Multimedia
- I/O Ports
- Data Storage
- CODEC$_5$
- General Purpose PCI Bus Interfacing

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

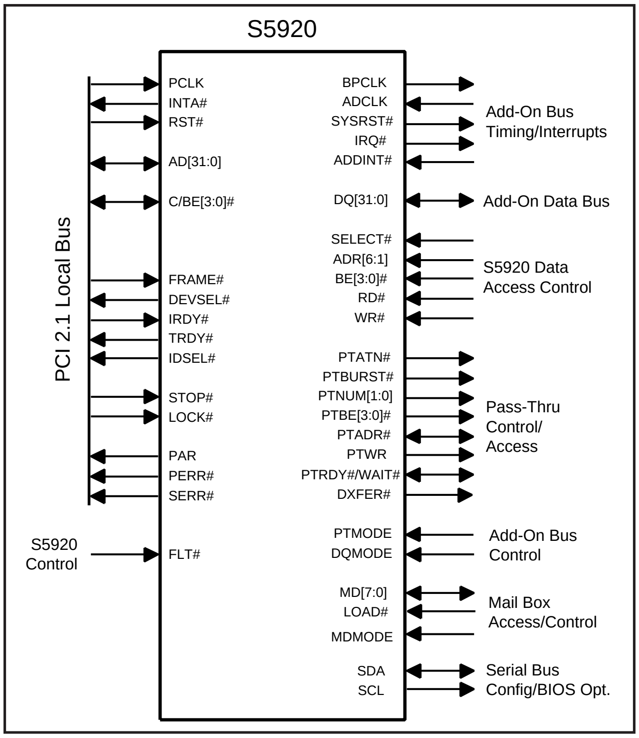

User Application
S5920
PCI Configuration Registers
2.1 PCI Local Bus Interface Logic
Mux/ Demux
Data Buffers
ADD-ON Pass-Thru
32 Byte FIFO
32 Byte FIFO
PCI Pass-Thru
Pass-Thru Address Register
Mailboxes/Status
Operation/ Status Registers
AMCC ADD-ON Local Bus Interface Logic
Mux/ Demux Active R/W Logic Buffers
ISA Card Design
ISDN FDDI ATM
Graphics/ MPEG/ Grabber
Audio Design
I/O Controller
PCI Local Bus
Serial Read/Write Control
Serial Read/Write Control
Configuration Space Expansion BIOS Serial NVRAM
Serial Bus

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.

Figure 2. S5920 Pinout



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.

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.

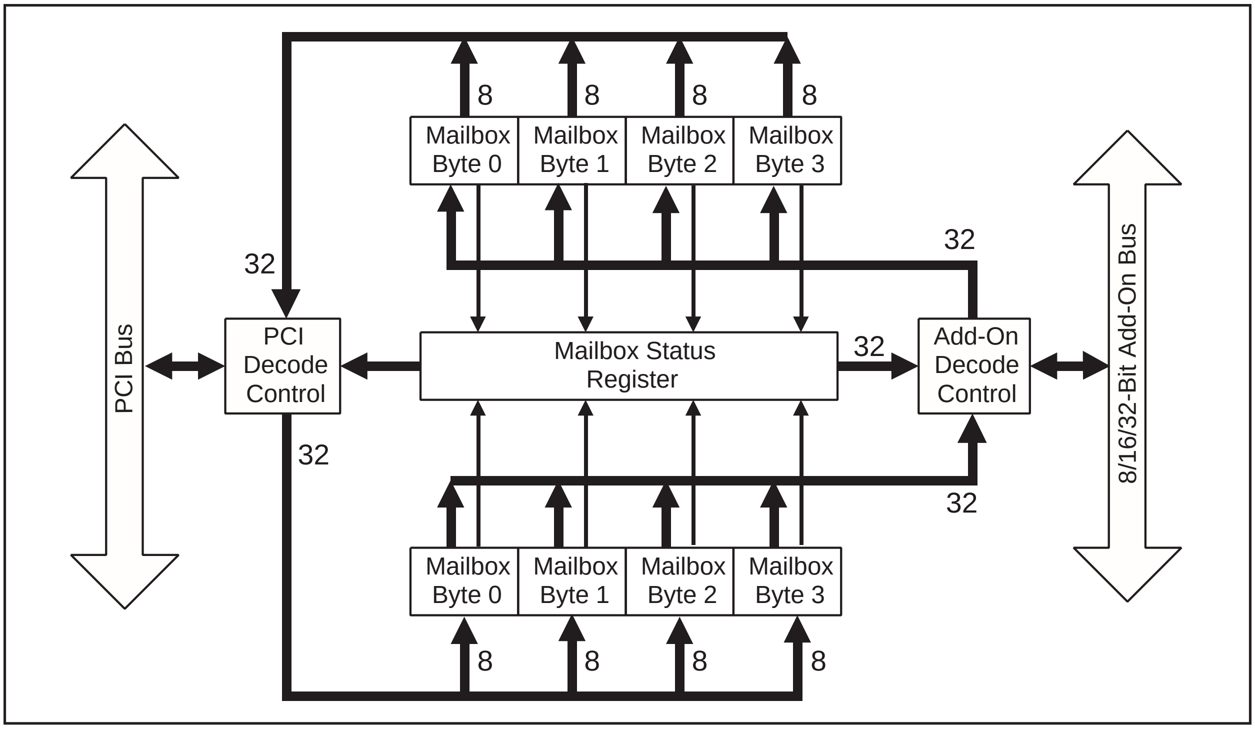
PASS-THRU OPERATION

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

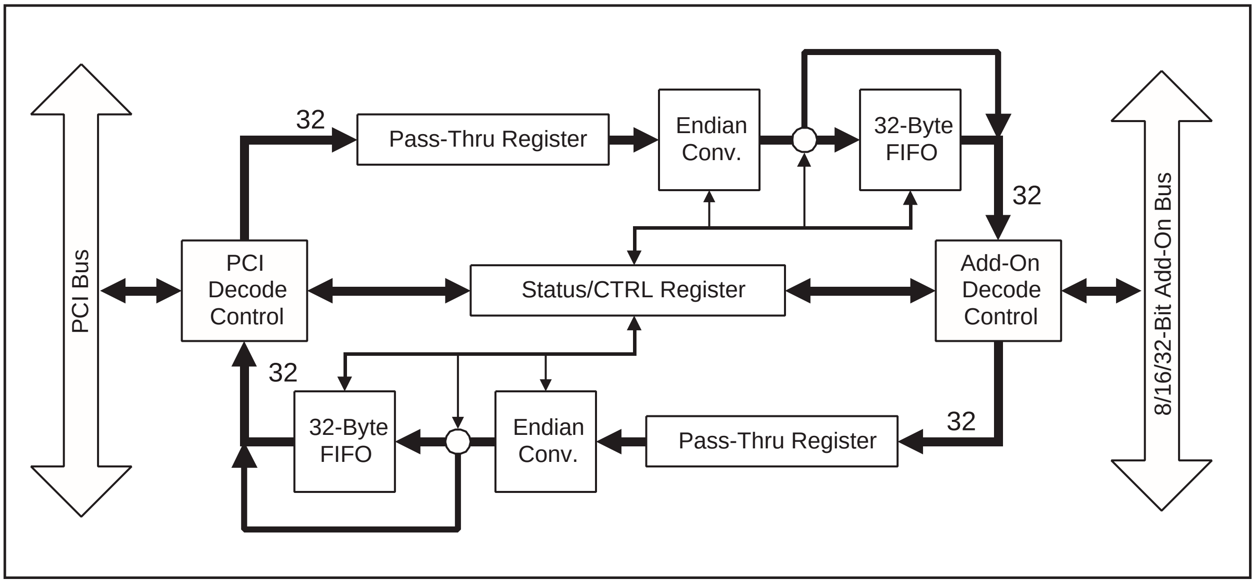


To increase data throughput, the Pass-Thru channel incorporates two 32-byte FIFOs. One FIFO is dedicated to PCI read data while the other is dedicated to PCI write data. Enabling the write FIFO allows the S5920 to accept zero wait state bursts from the PCI bus regardless of the Add-On bus application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

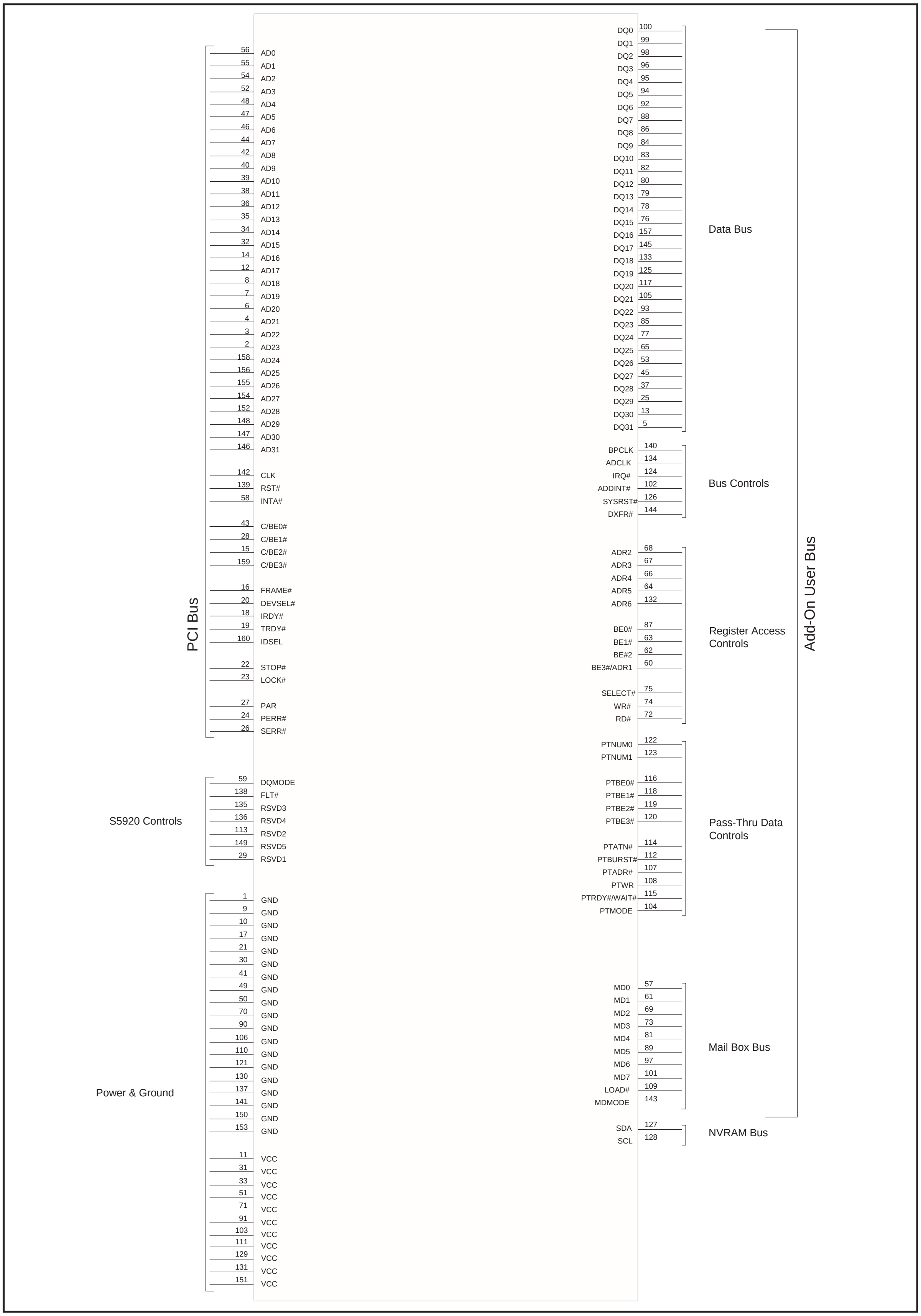
1010b
0110b

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.

PCI CONFIGURATION REGISTERS

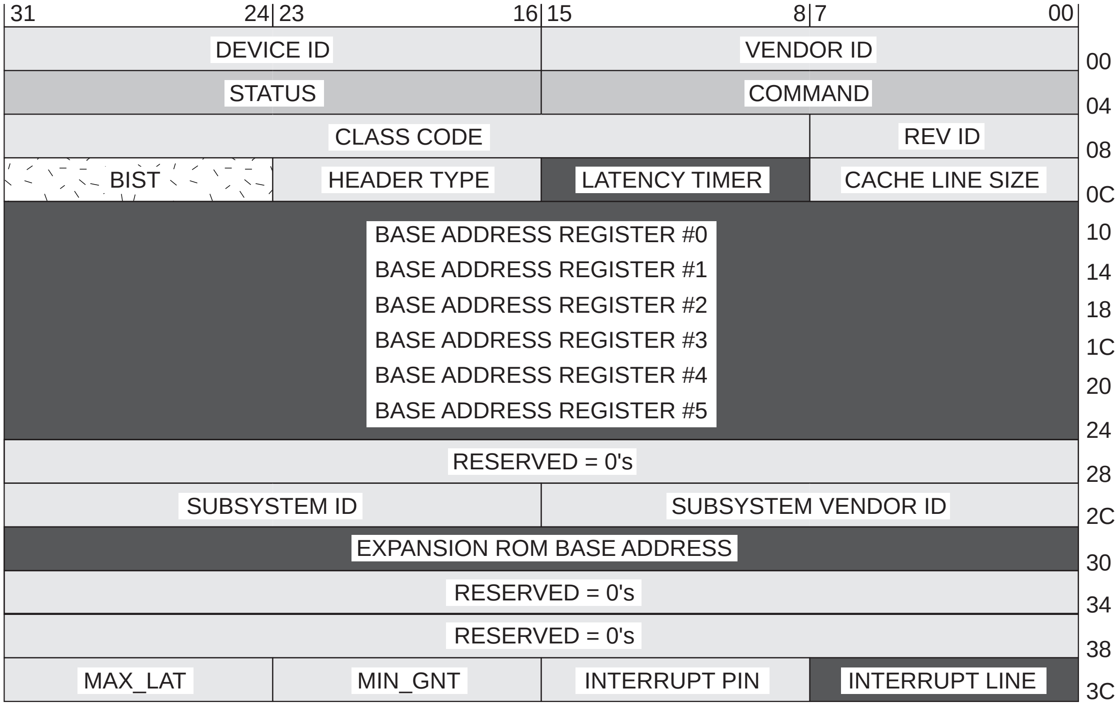
Each PCI bus device contains a unique 256-byte region called its configuration header space. Portions of this configuration header are mandatory in order for a PCI agent to be in full compliance with the PCI specification. This section describes each of the configuration space fields—its address, default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

Table 1. Configuration Registers

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



31 24	23 16	15 8	7 00	
DEVICE ID		VENDOR ID		00
STATUS		COMMAND		04
CLASS CODE			REV ID	08
BIST	HEADER TYPE	LATENCY TIMER	CACHE LINE SIZE	0C
BASE ADDRESS REGISTER #0				10
BASE ADDRESS REGISTER #1				14
BASE ADDRESS REGISTER #2				18
BASE ADDRESS REGISTER #3				1C
BASE ADDRESS REGISTER #4				20
BASE ADDRESS REGISTER #5				24
RESERVED = 0's				28
SUBSYSTEM ID		SUBSYSTEM VENDOR ID		2C
EXPANSION ROM BASE ADDRESS				30
RESERVED = 0's				34
RESERVED = 0's				38
MAX_LAT	MIN_GNT	INTERRUPT PIN	INTERRUPT LINE	3C

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

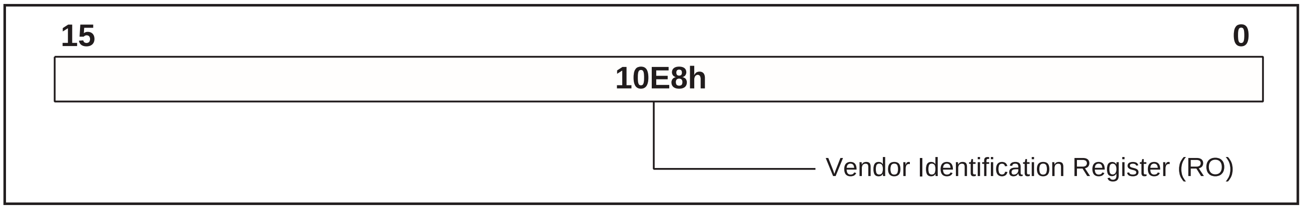
Note: Some registers are a combination of the above. See individual sections for full description.

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



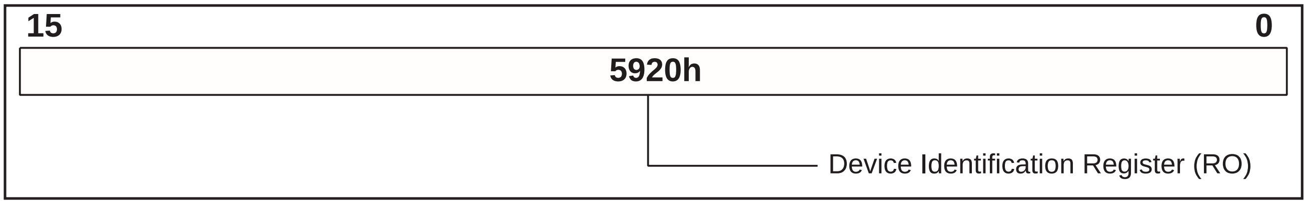
Bit	Description
15.0	Vendor Identification Number: AMCC's 16-bit value is 10E8h

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 nonvolatile device is used for boot loading.

Figure 2. Device Identification Register



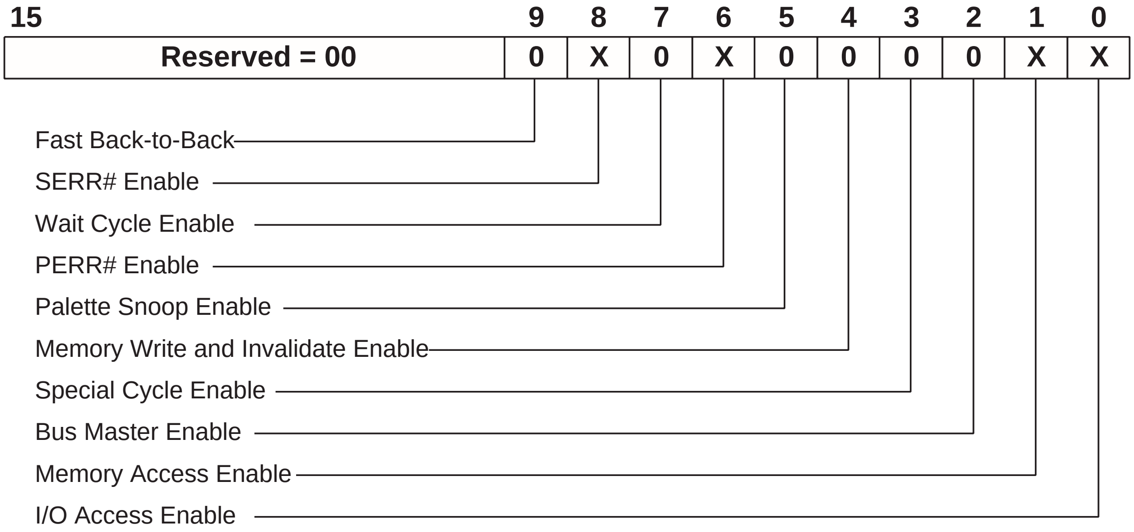
Bit	Description
15.0	Device Identification Number: AMCC's temporary end user value 5920h

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



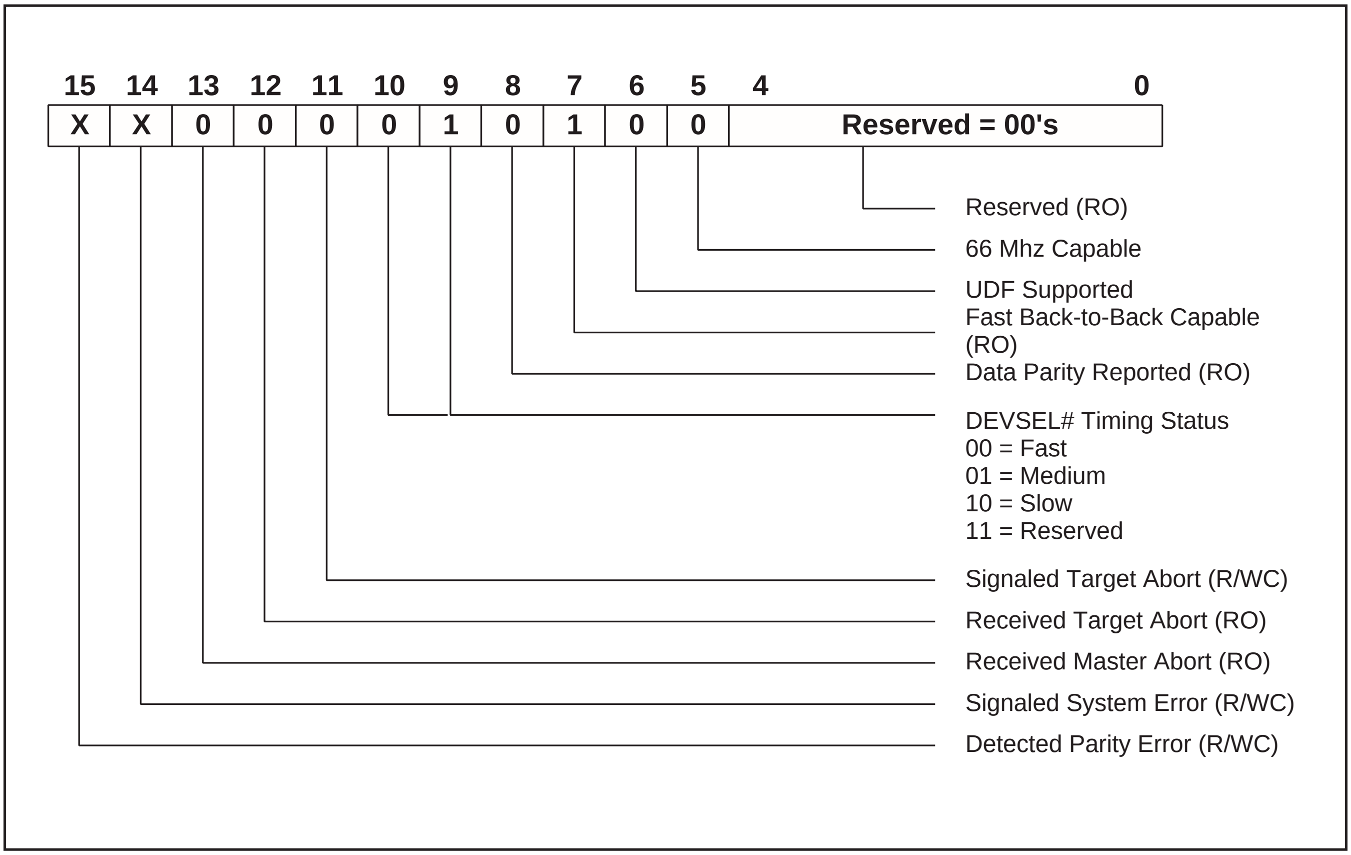
Bit	Description
15:10	Reserved. Hardwired to 0.
9	Fast Back-to-Back Enable. This bit enables fast back-to-back capability for bus master transaction. The S5920 is a target-only device and hardwires this bit to a 0.
8	System Error Enable. Setting this bit to a 1 allows the S5920 to drive the SERR# signal. 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



15
14
13
12
11
10
9
8
7
6
5
4
0
X
X
0
0
0
0
1
0
1
0
0
Reserved = 00's
Reserved (RO)
66 Mhz Capable
UDF Supported
Fast Back-to-Back Capable (RO)
Data Parity Reported (RO)
DEVSEL# Timing Status
00 = Fast
01 = Medium
10 = Slow
11 = Reserved
Signaled Target Abort (R/WC)
Received Target Abort (RO)
Received Master Abort (RO)
Signaled System Error (R/WC)
Detected Parity Error (R/WC)

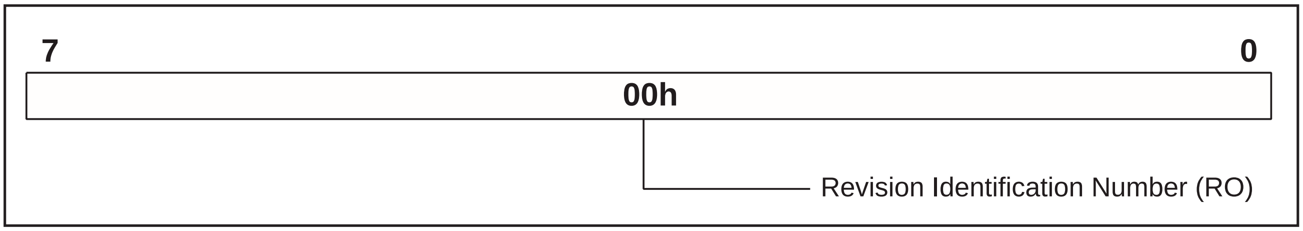
Bit	Description
15	Detected Parity Error. This bit is set whenever the S5920 detects a parity error. It is set independent of the state of Command Register Bit 6. The bit is cleared by writing a 1.
14	Signaled System Error. This bit is set whenever the S5920 generates the SERR# signal. This bit can be 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

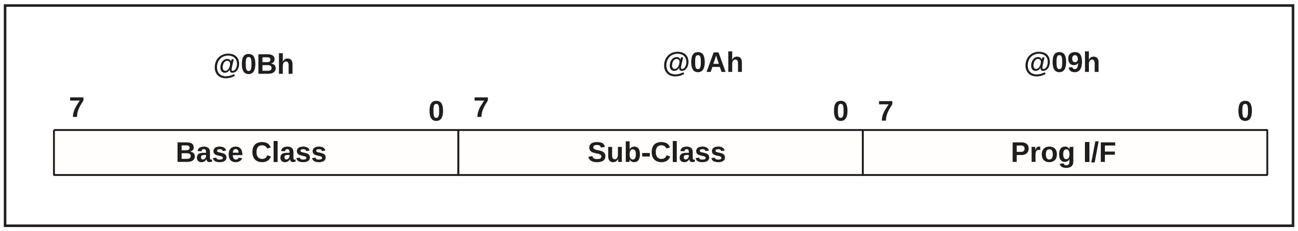


Table 2. Defined Base Class Codes

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

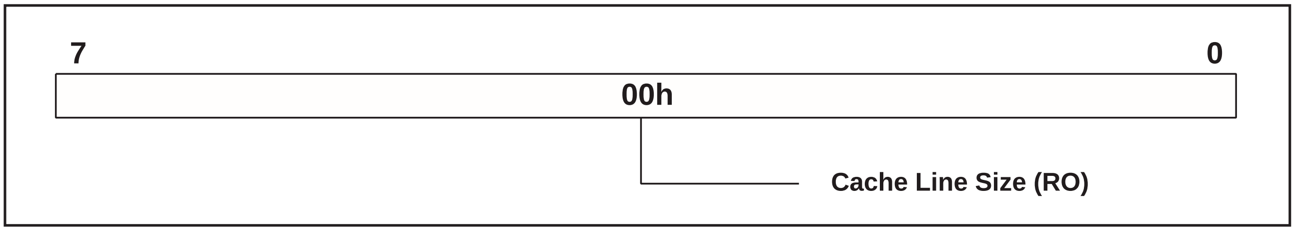
Sub-Class	Prog I/F	Description
00	00h	FireWire™ (IEEE 1394)
01h	00h	ACCESS.bus
02h	00h	SSA
03h	00h	Universal Serial Bus
04h	00h	Fibre Channel

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

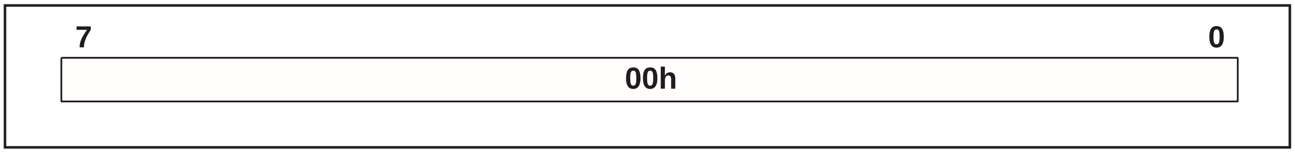


LATENCY TIMER REGISTER (LAT)

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

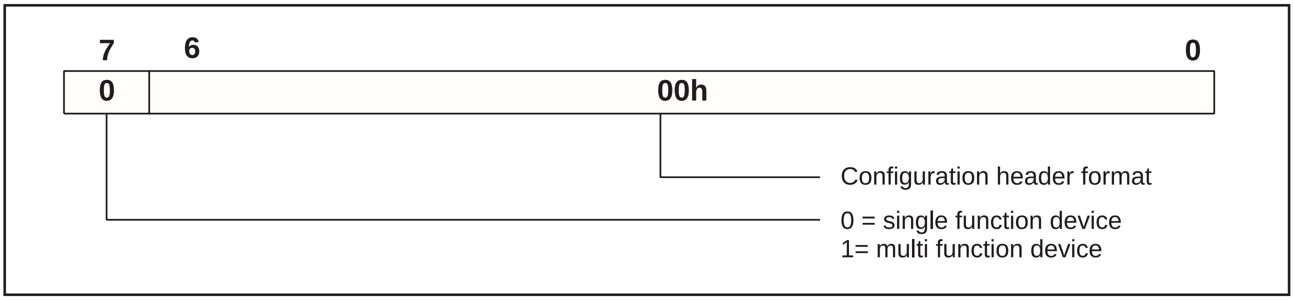


HEADER TYPE REGISTER (HDR)

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

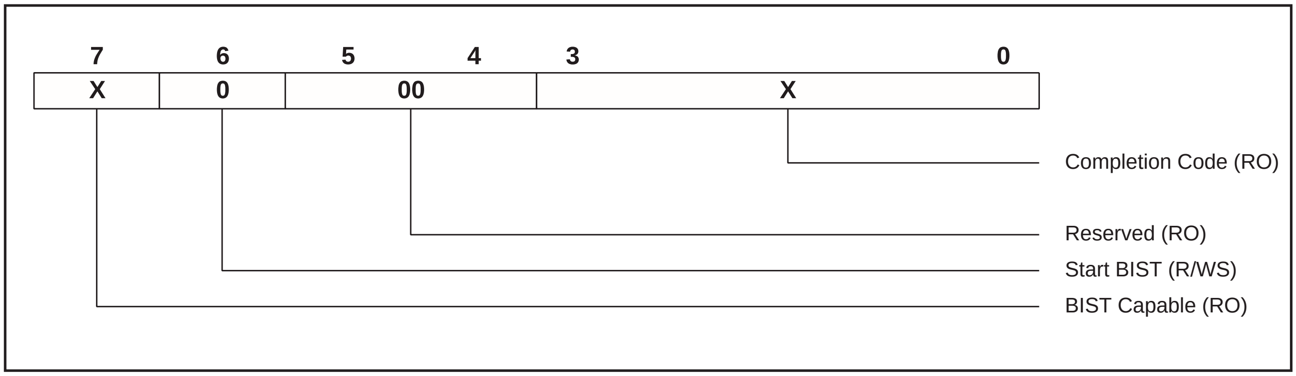


BUILT-IN SELF-TEST REGISTER (BIST)

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.

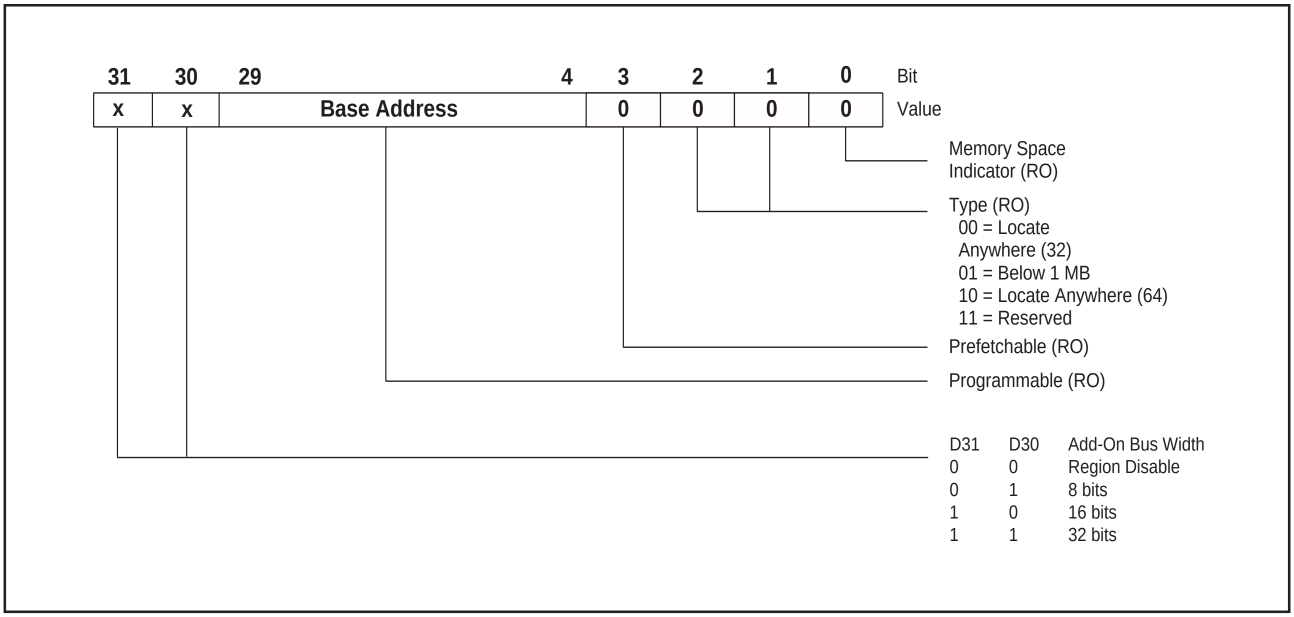
Determining Base Address Size

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

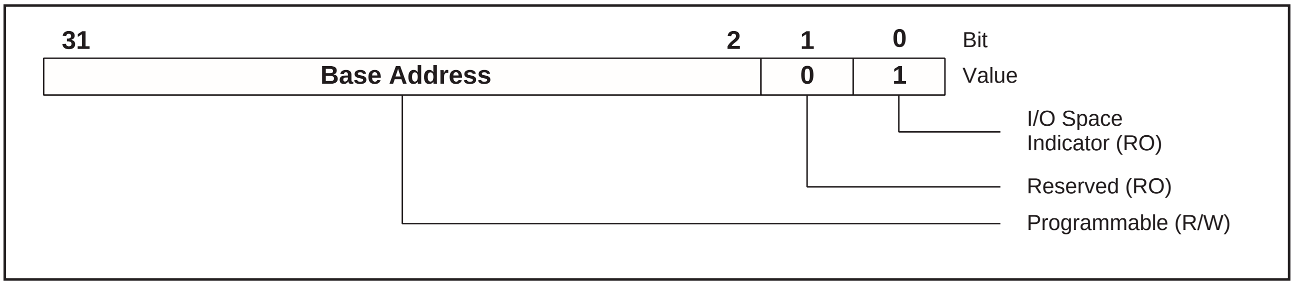
After a base address has been sized, the BIOS can physically locate it in memory (or I/O) space. The base address value must be on a natural binary boundary for the required size. For example, the first base address register returns FFFFFF81h indicating an I/O space (D0=1) of size 80h. This means that 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

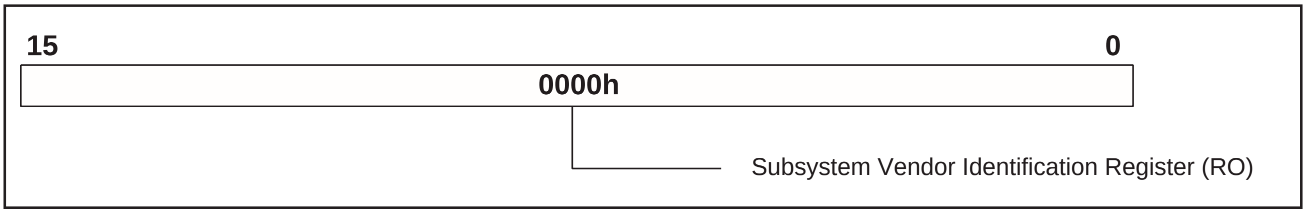
[1] Base Address Register 0, at offset 10h, powers up as FFFFFF81h. This default assignment allows usage without an external boot memory. Should an nvRAM be used, the base address can be boot loaded to become a memory space (FFFFFF80h or FFFFFF82h).

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



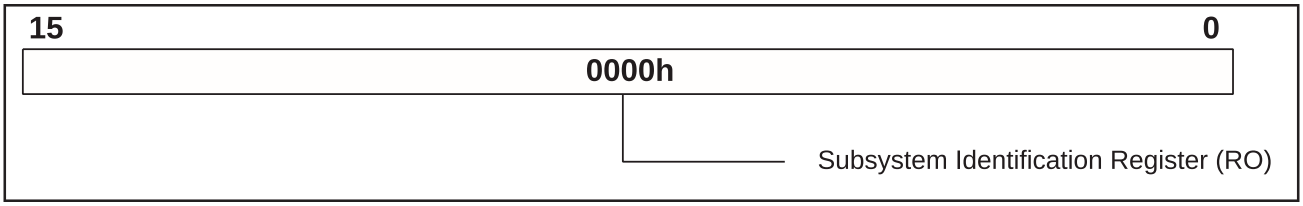
Bit	Description
15:0	Subsystem Vendor Identification Number.

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



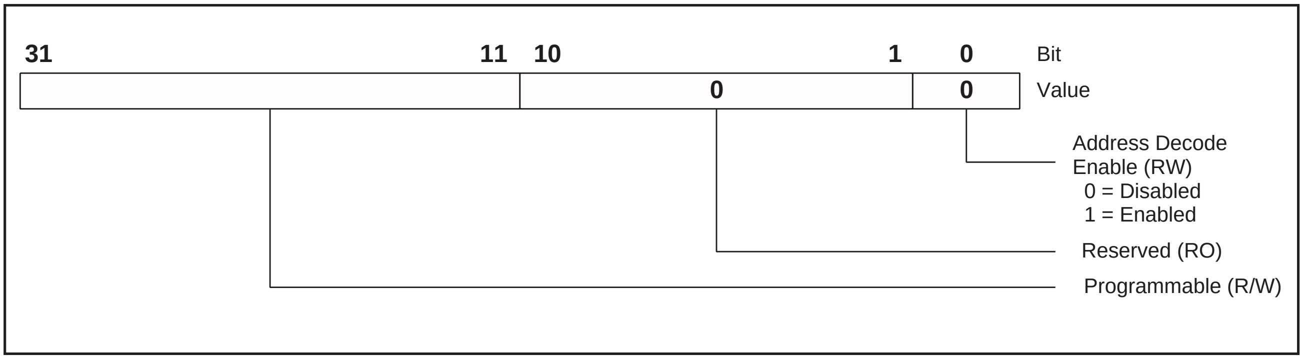
Bit	Description
15:0	Subsystem Identification Number.

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

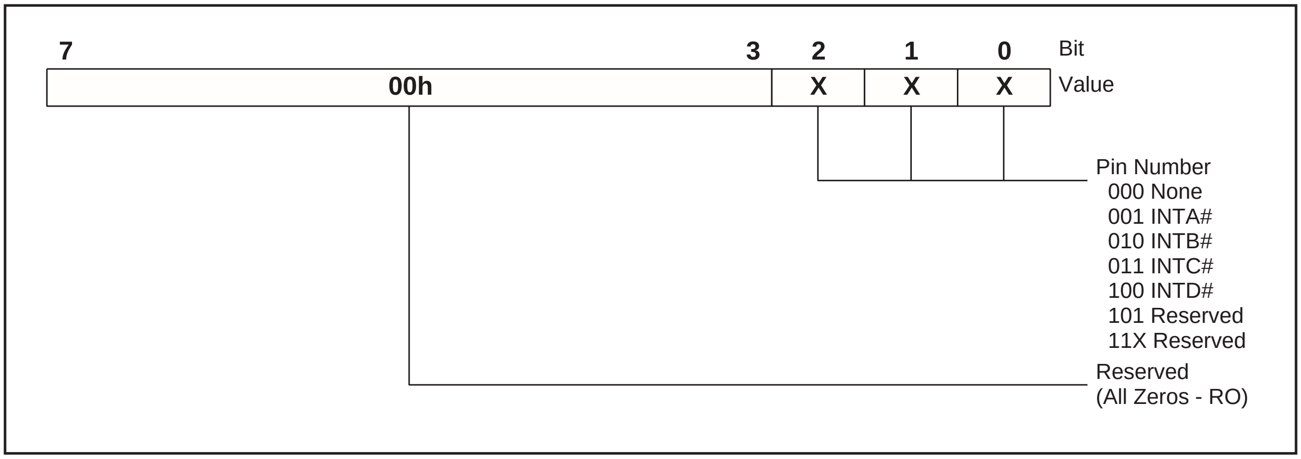
7	0
FFh	

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

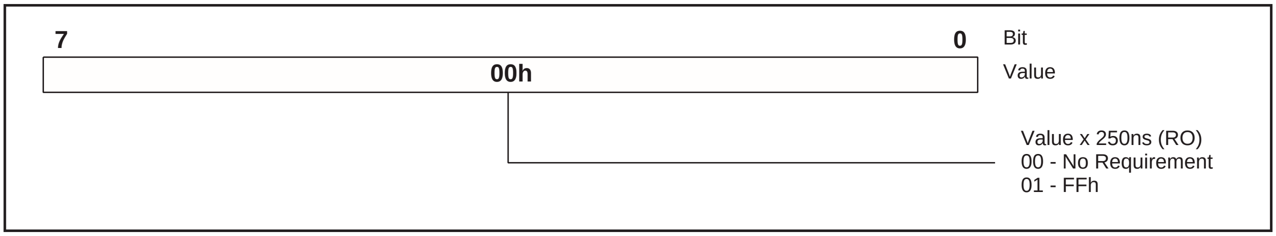


MINIMUM GRANT REGISTER (MINGNT)

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

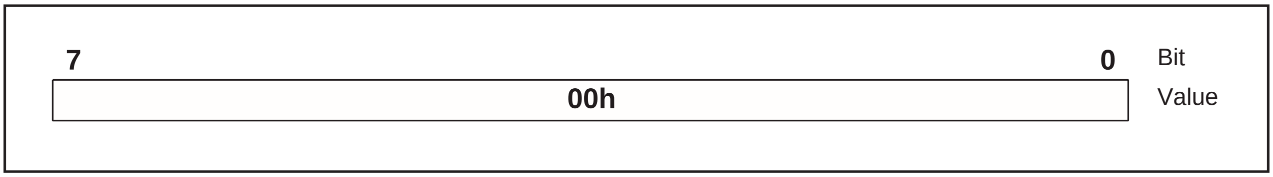


MAXIMUM LATENCY REGISTER (MAXLAT)

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

31 24	23 16	15 8	7 0
Byte 3	Byte 2	Byte 1	Byte 0

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

31 24	23 16	15 8	7 0
Byte 3	Byte 2	Byte 1	Byte 0

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)

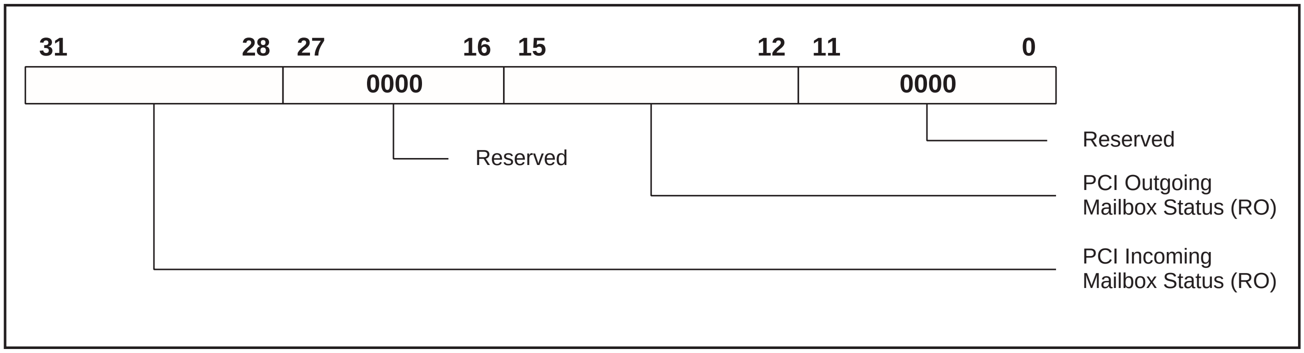

31
28
27
16
15
12
11
0
0000
0000
Reserved
Reserved
PCI Outgoing Mailbox Status (RO)
PCI Incoming Mailbox Status (RO)

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

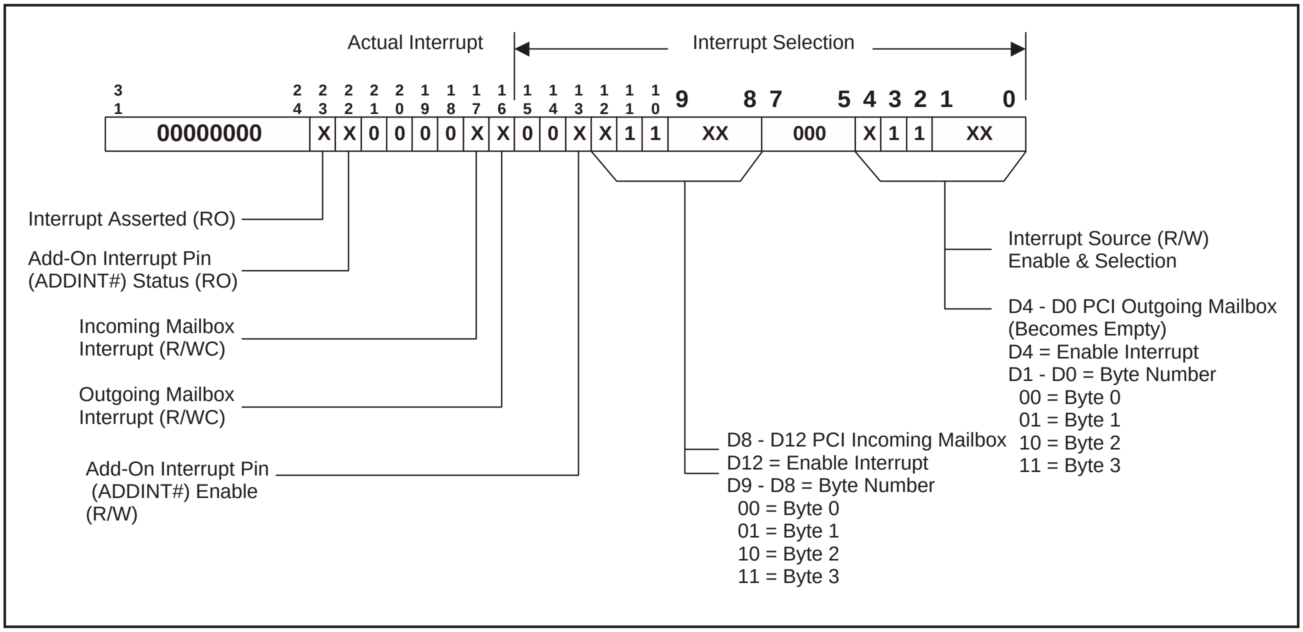

Actual Interrupt
Interrupt Selection
31 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 5 4 3 2 1 0
00000000 X X 0 0 0 0 X X 0 0 X X 1 1 XX 000 X 1 1 XX
Interrupt Asserted (RO)
Add-On Interrupt Pin (ADDINT#) Status (RO)
Incoming Mailbox Interrupt (R/WC)
Outgoing Mailbox Interrupt (R/WC)
Add-On Interrupt Pin (ADDINT#) Enable (R/W)
D8 - D12 PCI Incoming Mailbox
D12 = Enable Interrupt
D9 - D8 = Byte Number
00 = Byte 0
01 = Byte 1
10 = Byte 2
11 = Byte 3
Interrupt Source (R/W) Enable & Selection
D4 - D0 PCI Outgoing Mailbox (Becomes Empty)
D4 = Enable Interrupt
D1 - D0 = Byte Number
00 = Byte 0
01 = Byte 1
10 = Byte 2
11 = Byte 3

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

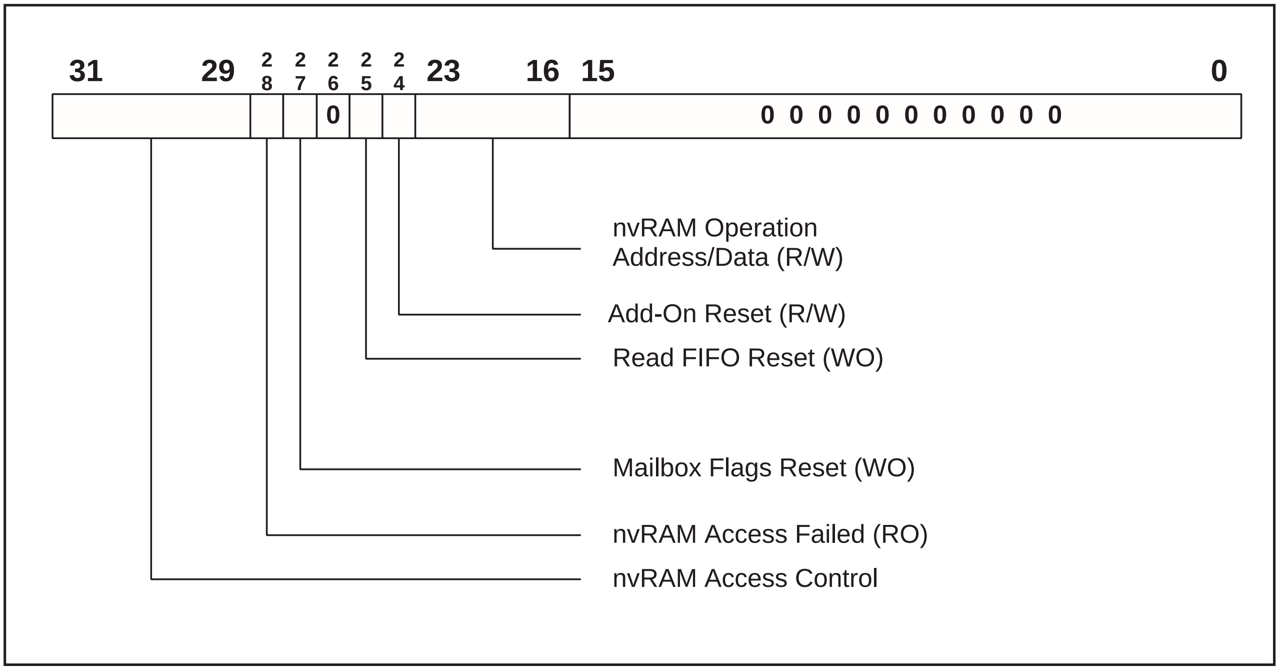


Table 4. Reset Control Register

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

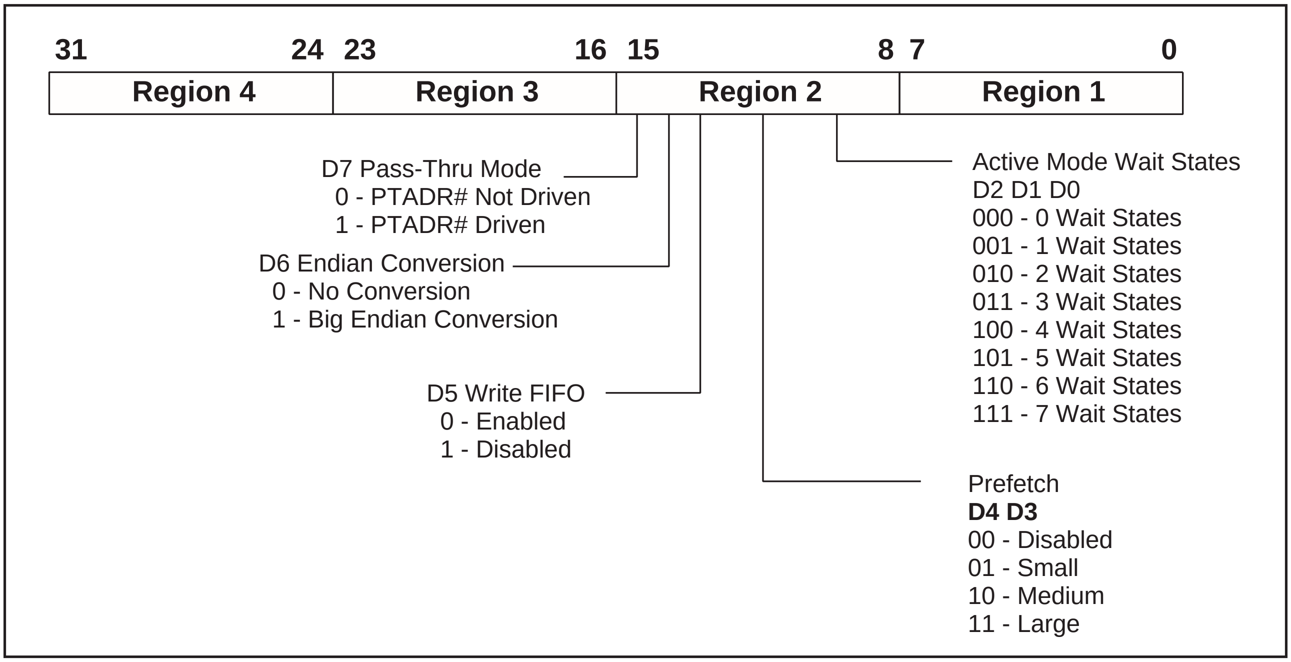


Table 5 describes one of the four configuration registers. All four region configuration registers are exactly the same.

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

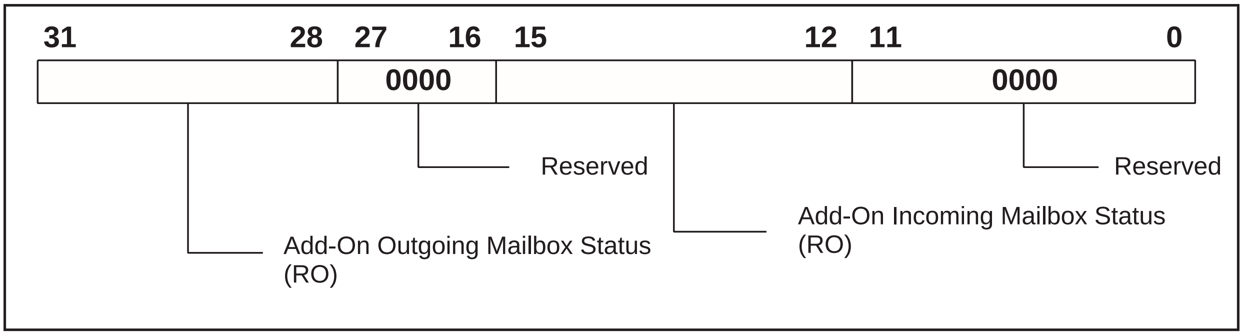

31
28
27
16
15
12
11
0
0000
0000
Reserved
Reserved
Add-On Incoming Mailbox Status (RO)
Add-On Outgoing Mailbox Status (RO)

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

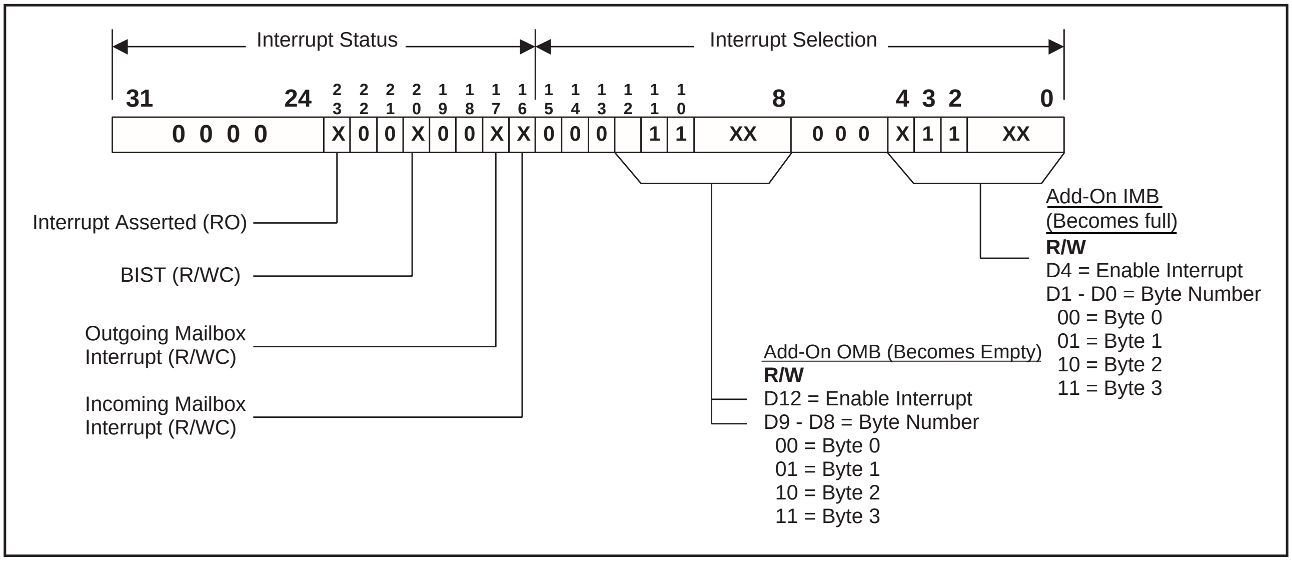


Table 8. Interrupt Control Status Register

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

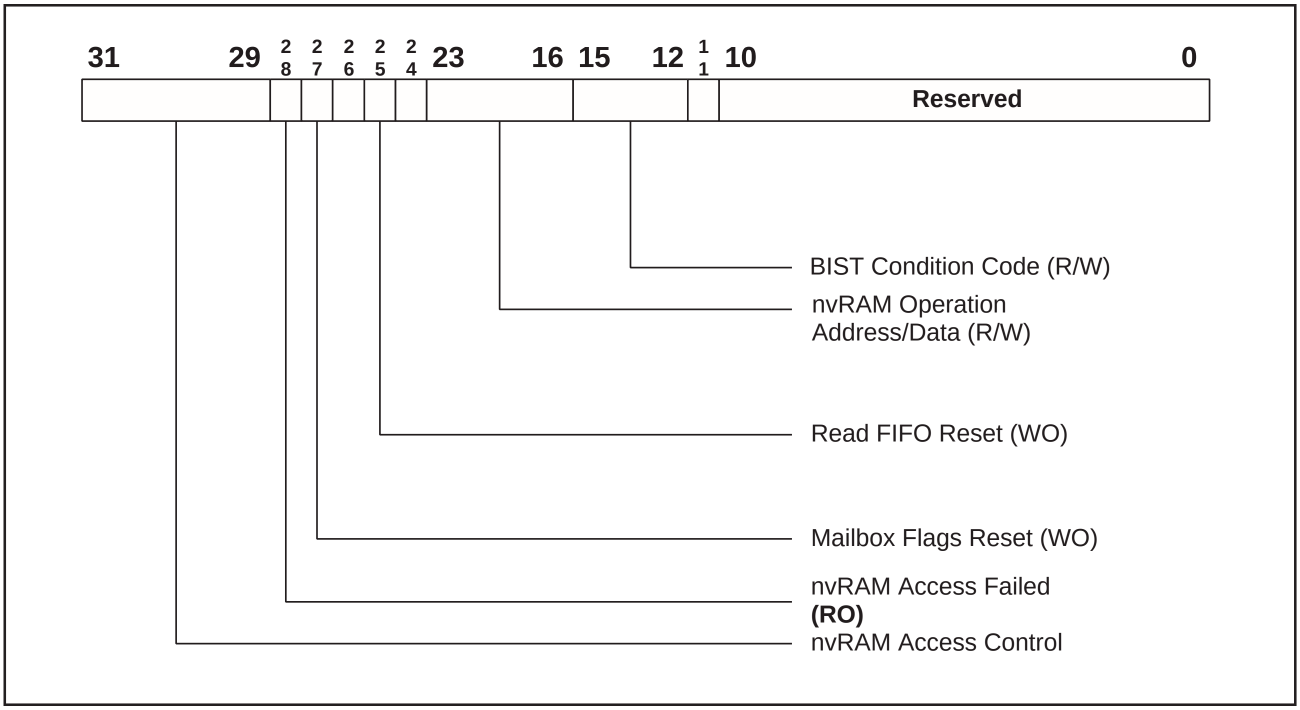


Table 9. Reset General Control/Status Register

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)

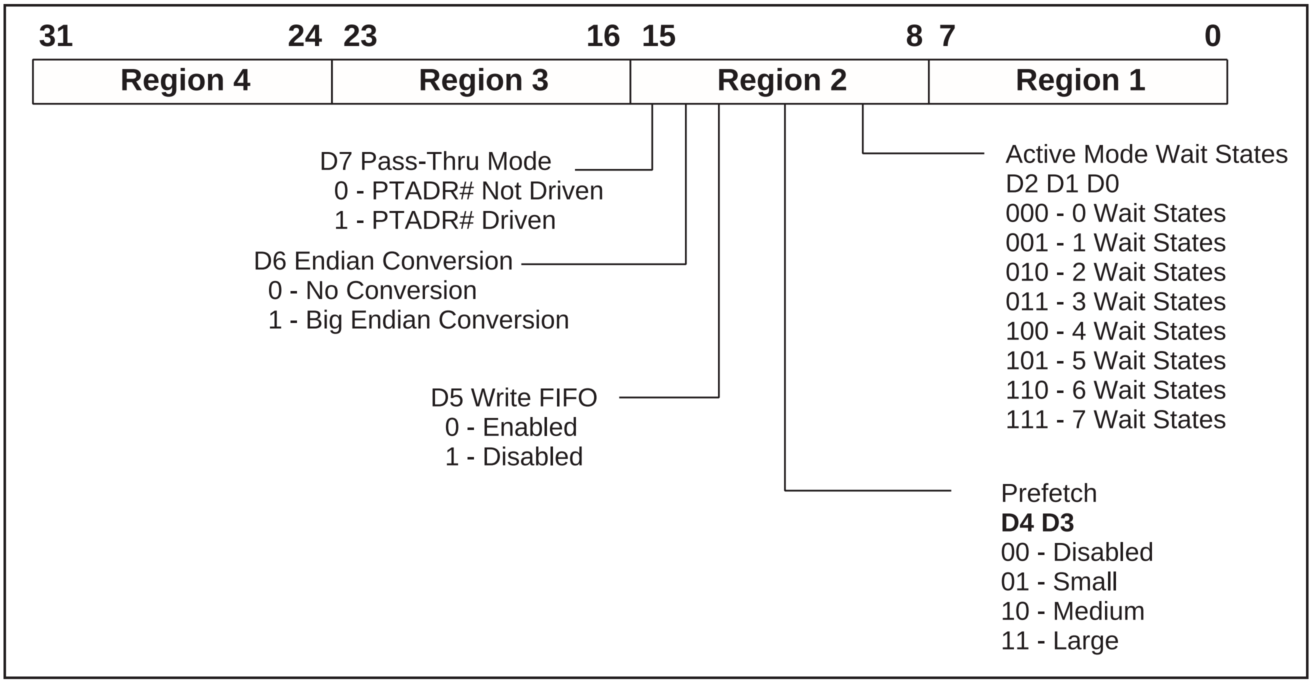
Register Name:	Pass-Thru Configuration Register
Add-On Address:	60h
Power-up value:	80808080h
Add-On Attribute:	Read/Write
Size:	32 bits

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

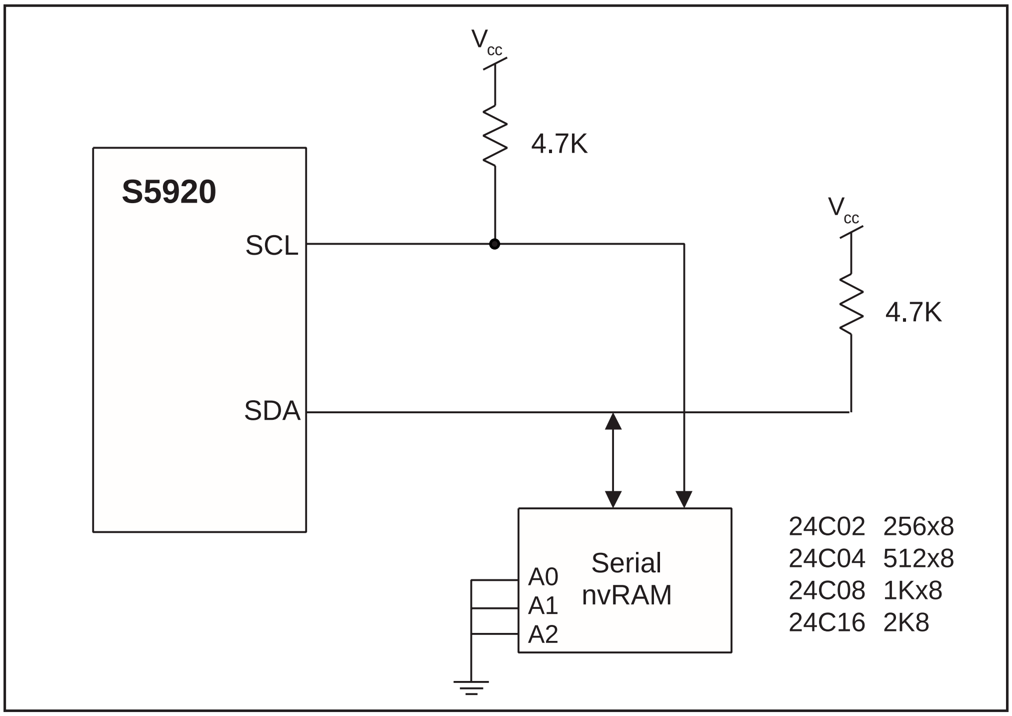


Figure 2. Serial Interface Definition of Start and Stop

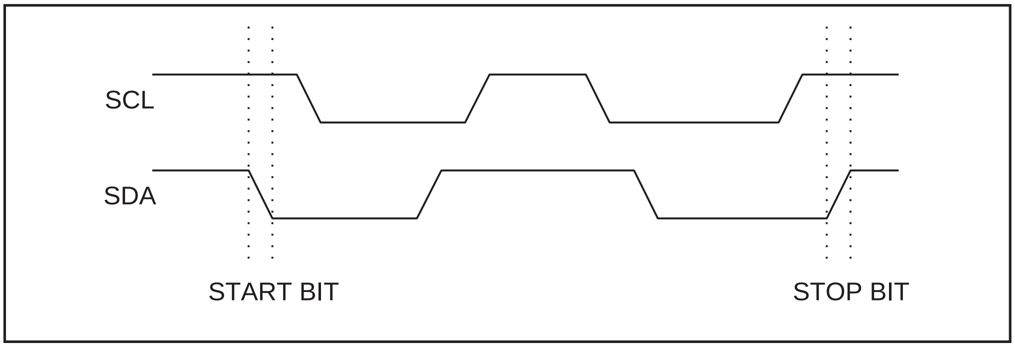


Figure 3. Serial Interface Clock Data Relationship

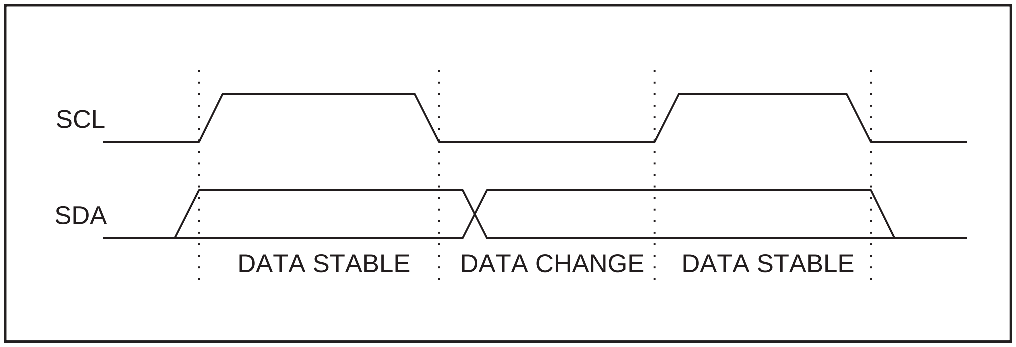


Figure 4. Serial Interface Byte Access-Write

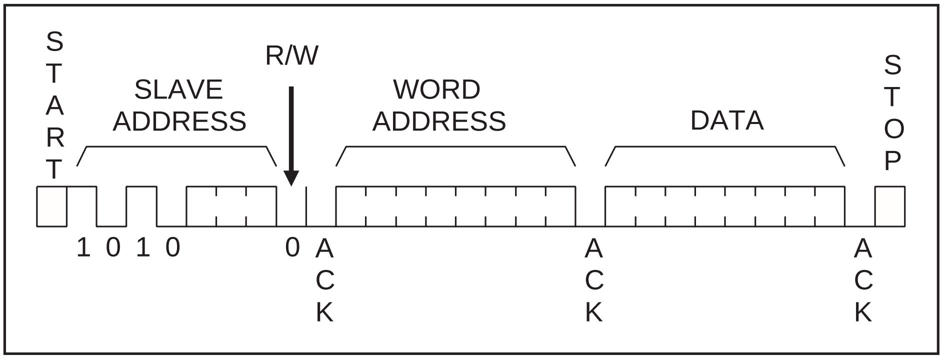


Figure 5. Serial Interface Byte Access-Read

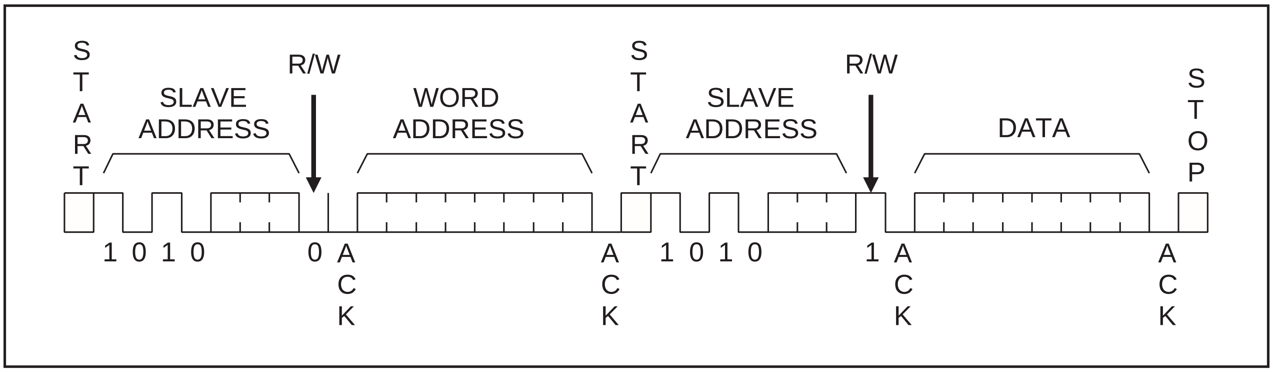
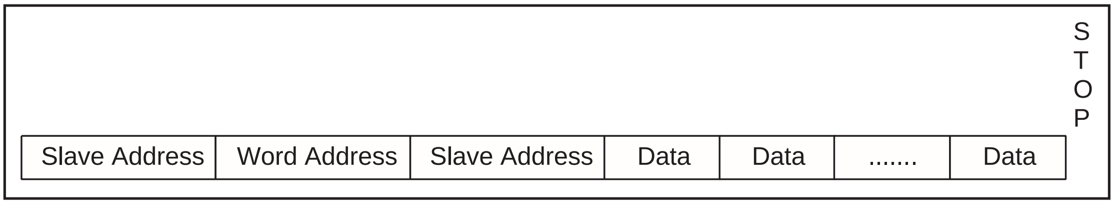


Figure 6. Serial Byte Access- Sequential Read



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

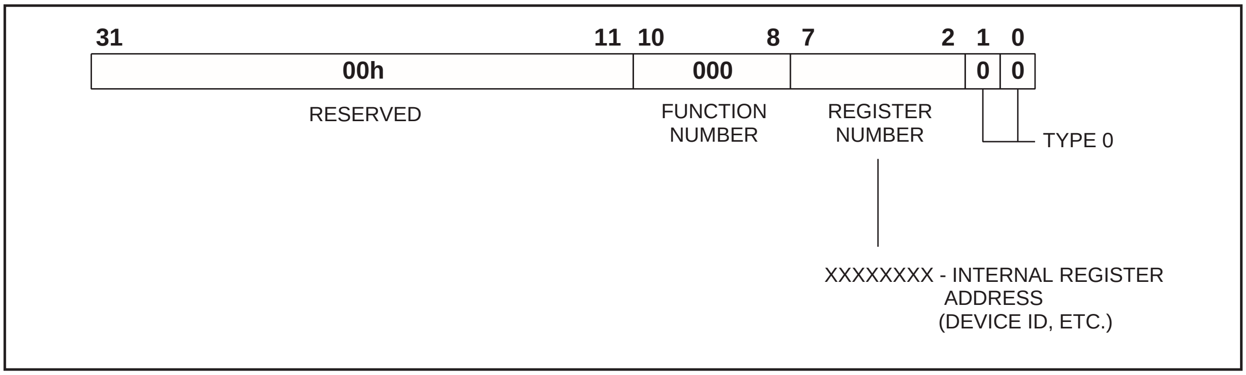
transfer to the nvRAM did not receive an ACKNOWLEDGE. The read data in RCR(32:16) should not be considered valid. This flag remains set until the start of the next read/write operation.

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

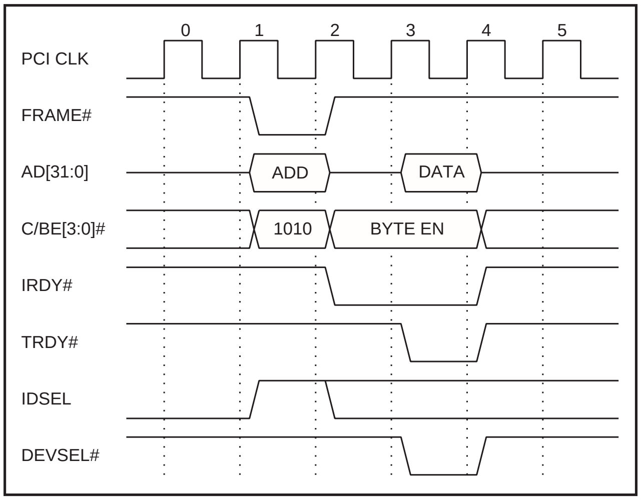
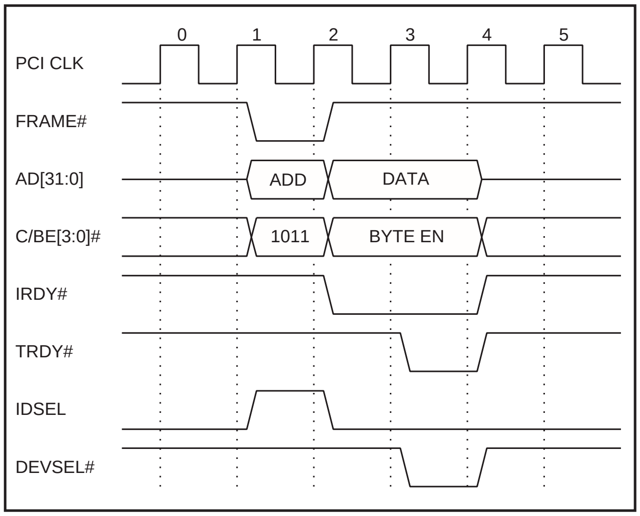


Figure 9. Type 0 Configuration Write Cycles



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 —			application specific	
1FFh,				
2FFh				
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.

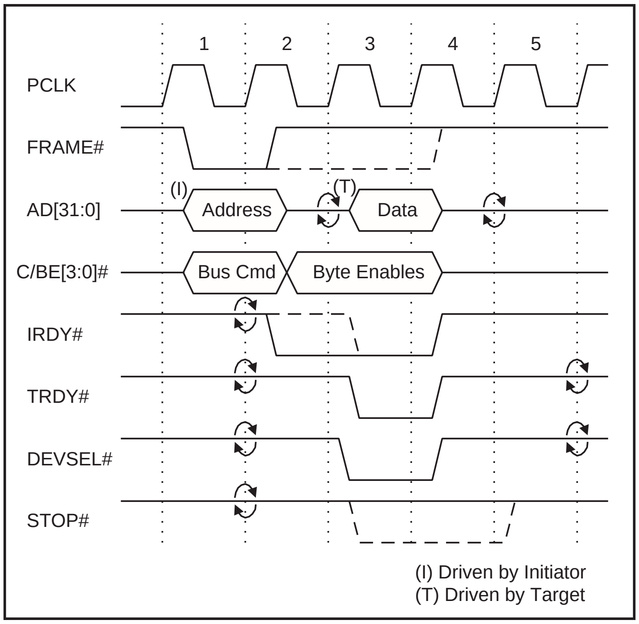
PCI READ TRANSFERS

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

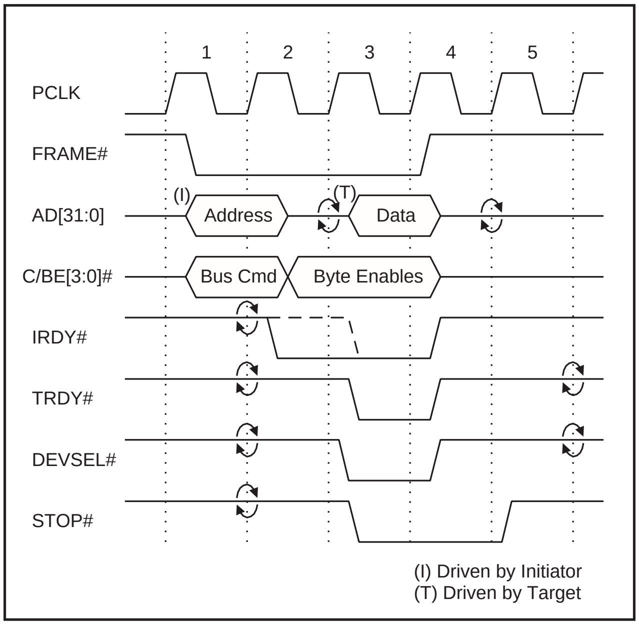
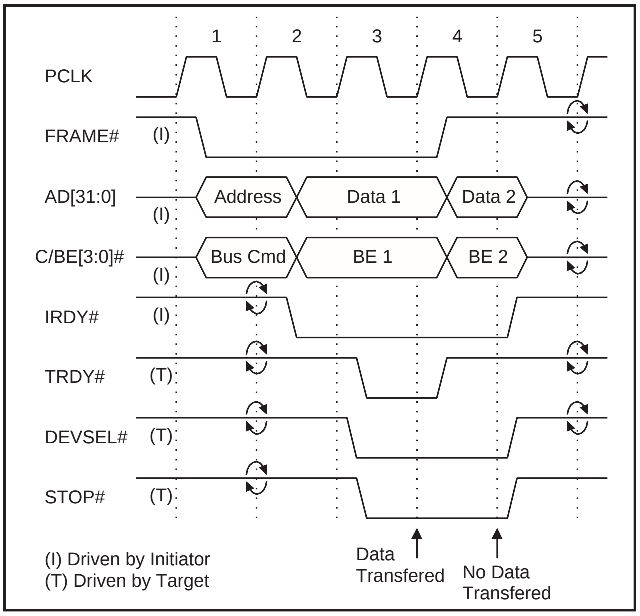


Figure 3. Burst PCI Bus Write of S5920 Registers



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)

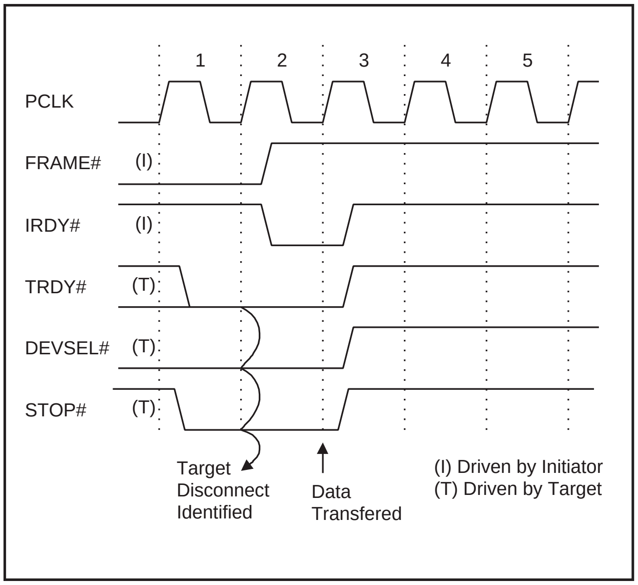
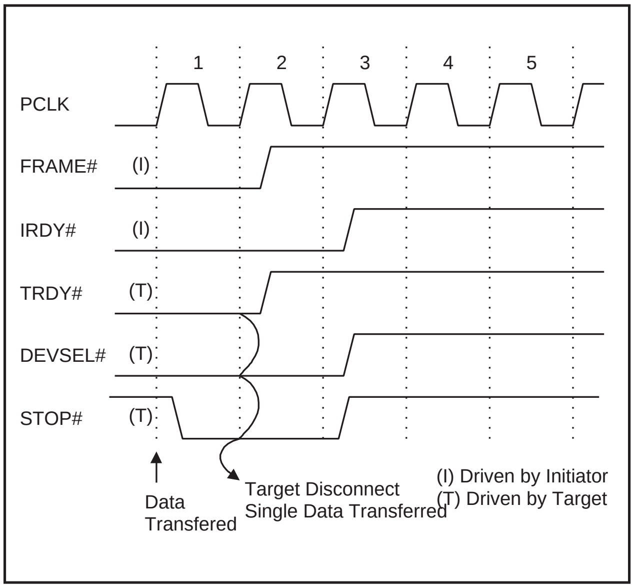


Figure 4b. Target Disconnect Example 2 (IRDY# Asserted)



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

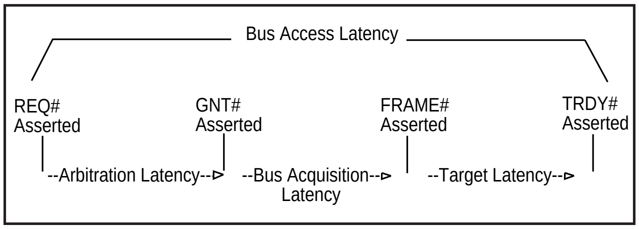


Figure 5. Target-Initiated Retry

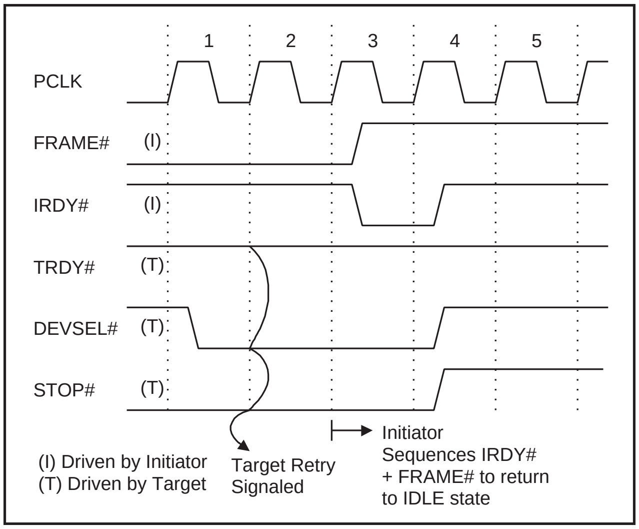


Figure 6. Engaging the LOCK# Signal

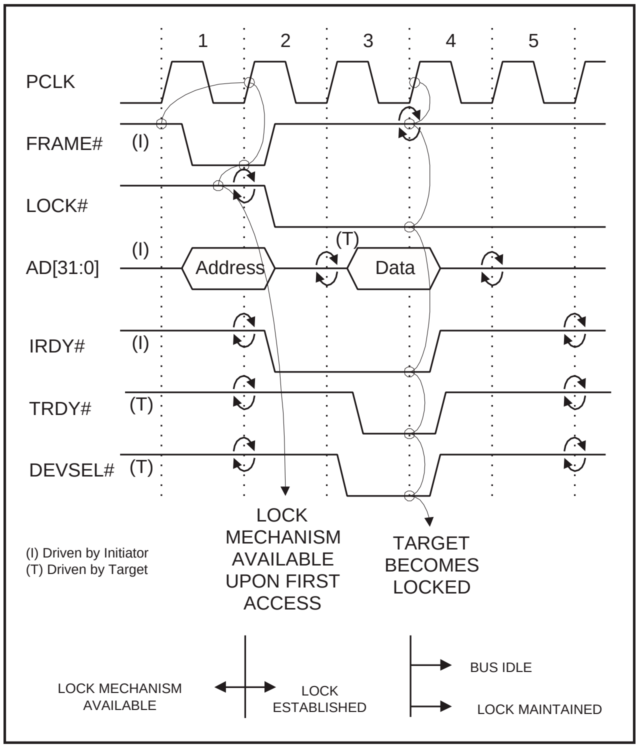


Table 2. Target Termination Type

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

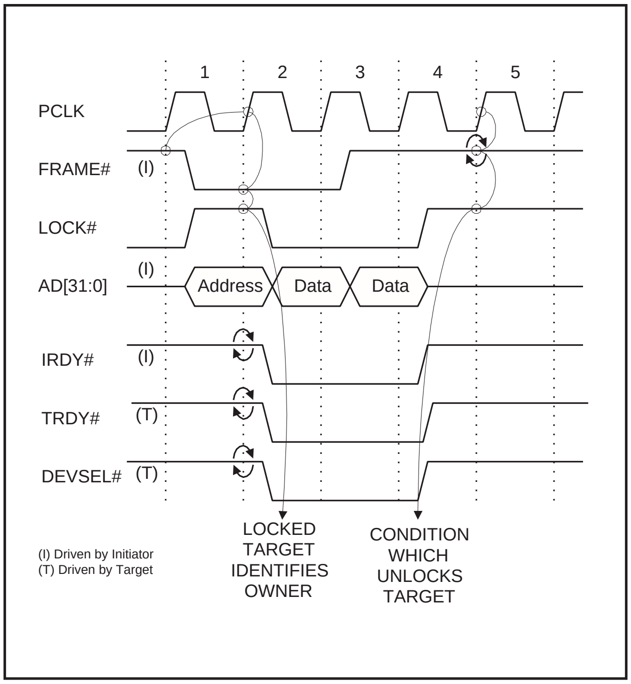


Figure 8. Access Attempt to a Locked Target

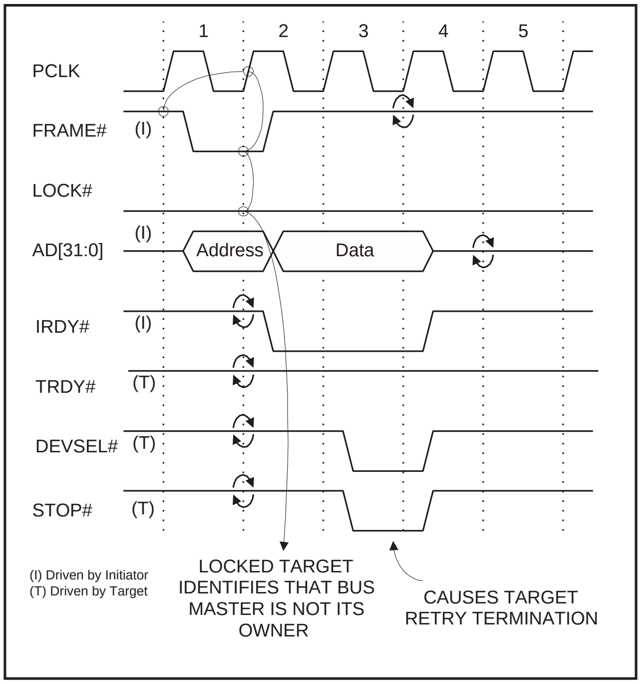
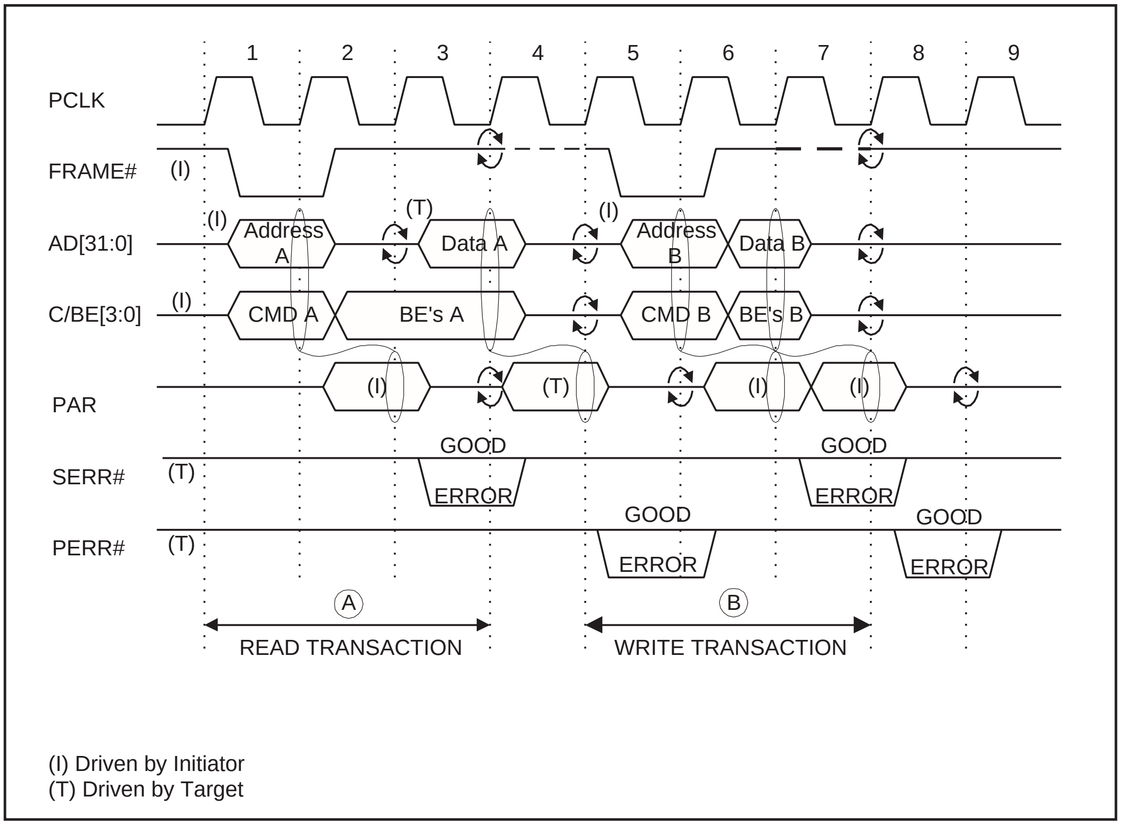


Figure 9. Error Reporting Signal



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

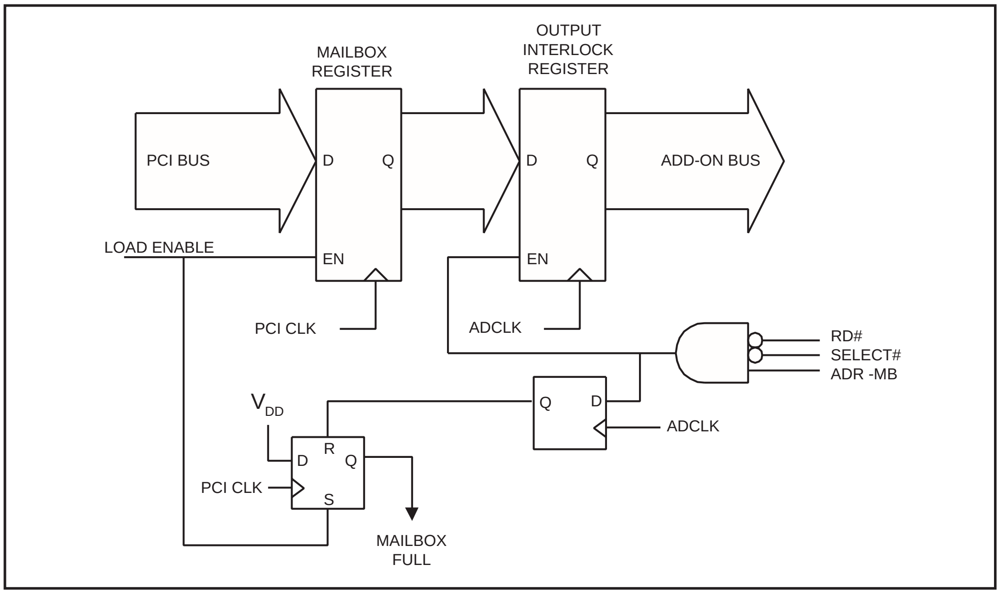
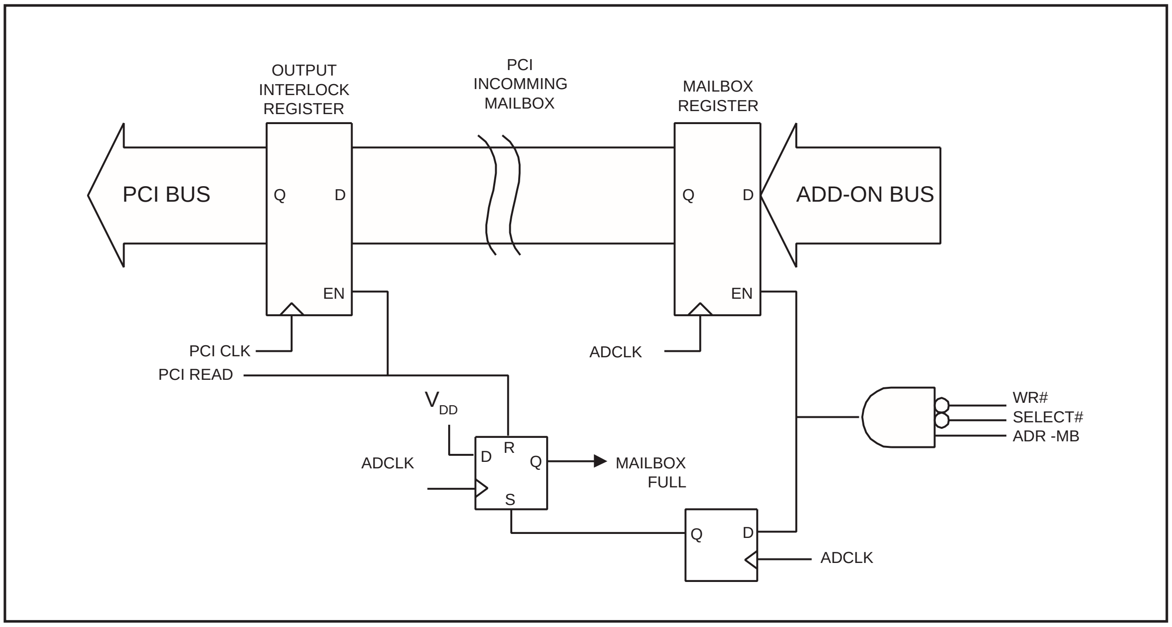


Figure 2. Add-On to PCI Mailbox Register



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)

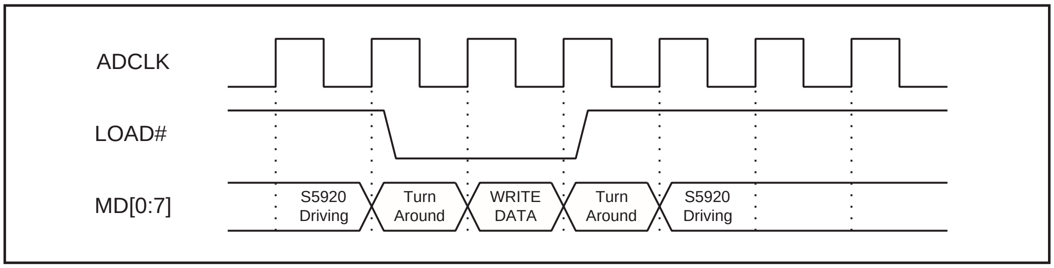
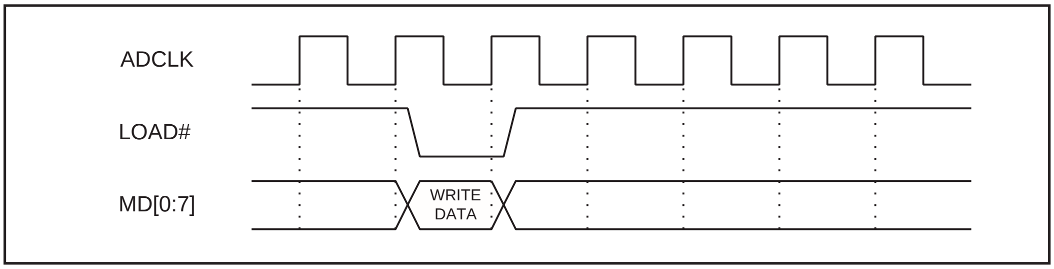


Figure 4. Input Mode (MDMODE=1)



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

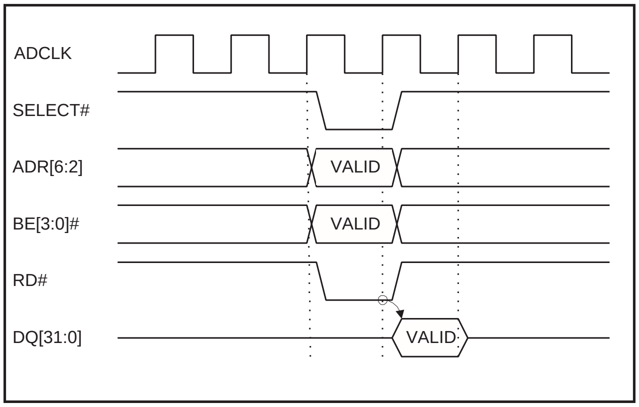
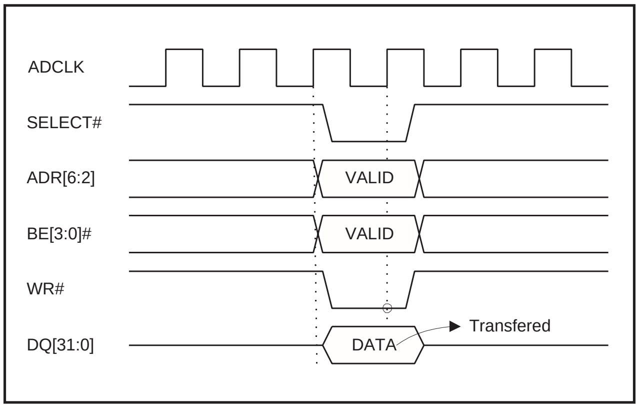


Figure 2. Write Operation Register



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

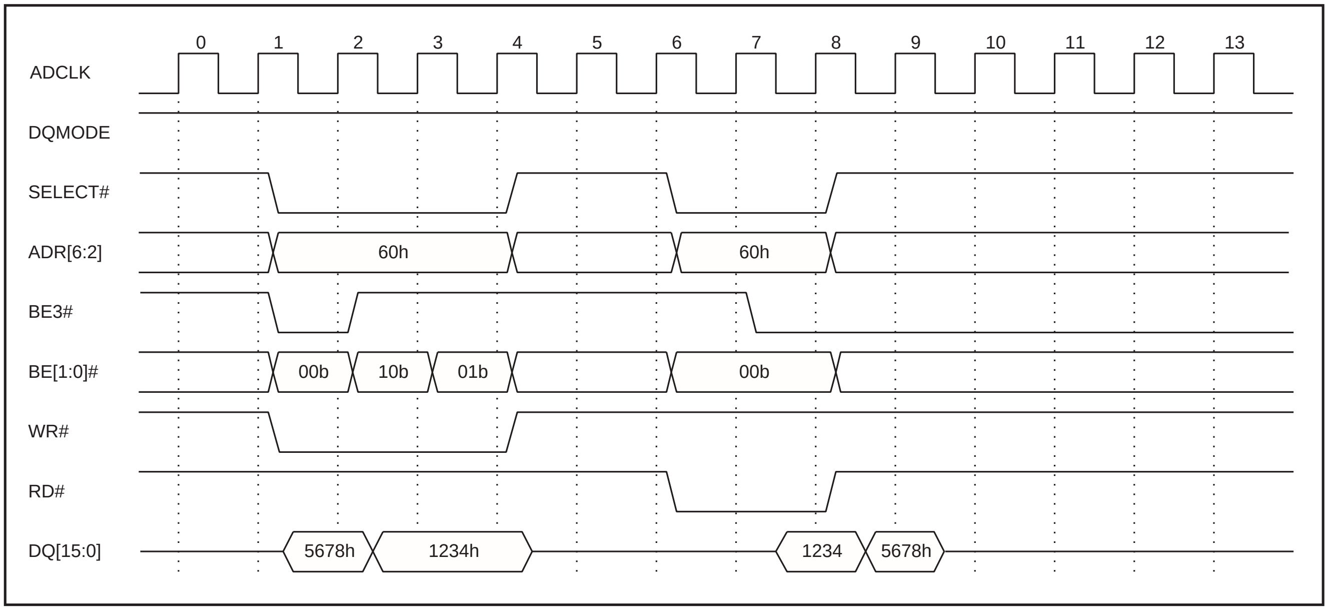


Figure 3 shows an example of a DWORD write of 12345678h using a 16 bit-mode write transfer, and is described as follows:

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.

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 4. PCI To Add-On Passive Write

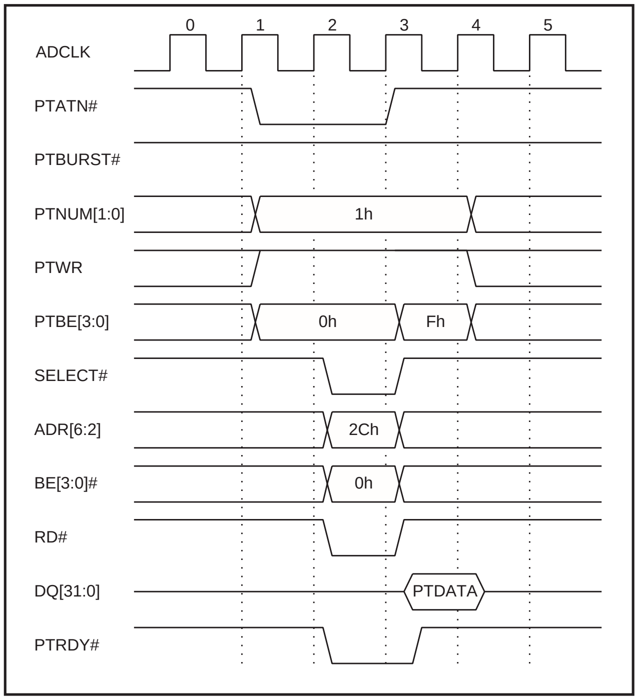
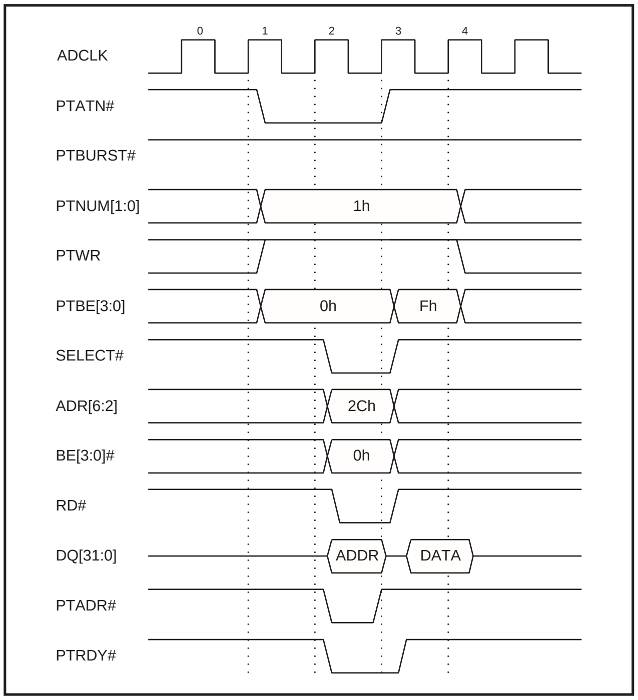


Figure 5. PCI To Add-On Passive Write w/Pass-Thru Address



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.

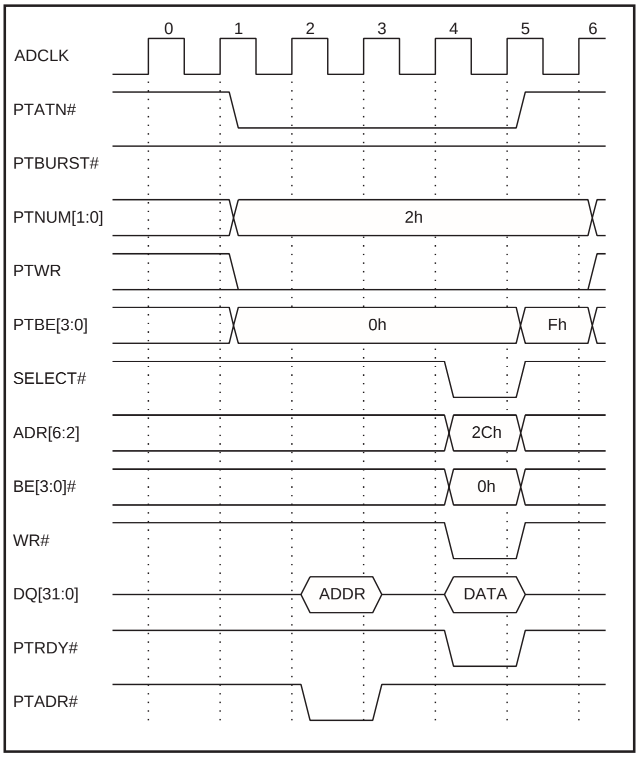
Clock 4: As PTATN# is deasserted, the Pass-Thru access is complete, and the S5920 can accept new Pass-Thru accesses starting on the next clock. The other Pass-Thru signals can also change (in anticipation of a new transfer). The S5920 stops driving the DQ bus as RD# and SELECT# were not valid on the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

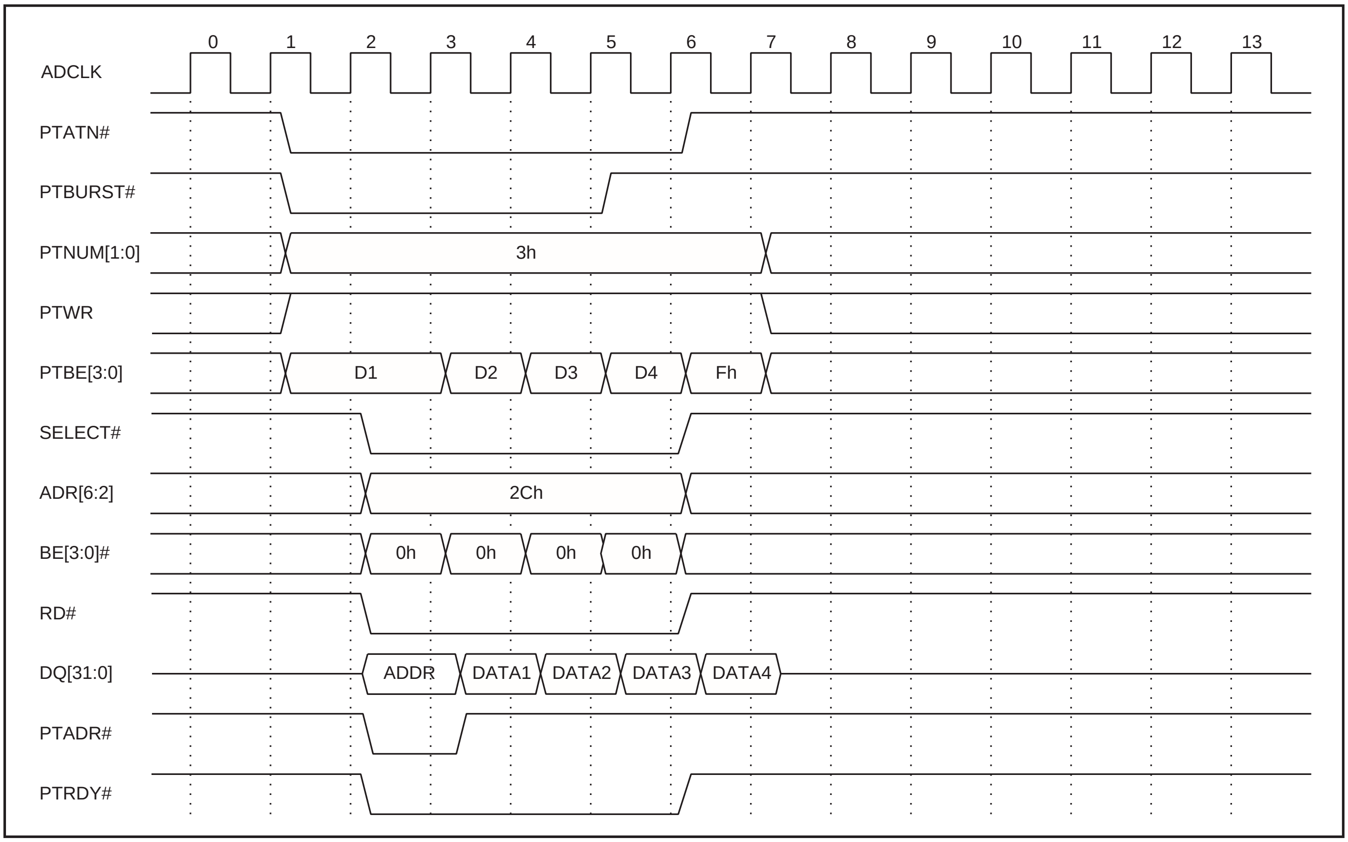


Figure 7 shows a Passive mode PCI to Add-On burst write of four DWORDs. In the following example, Add-On logic incorporates the use of PTADR# followed by multiple data reads to the S5920. If Add-On logic does not support burst accesses, PTADR# can be pulsed for individual data reads. The S5920 automatically 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

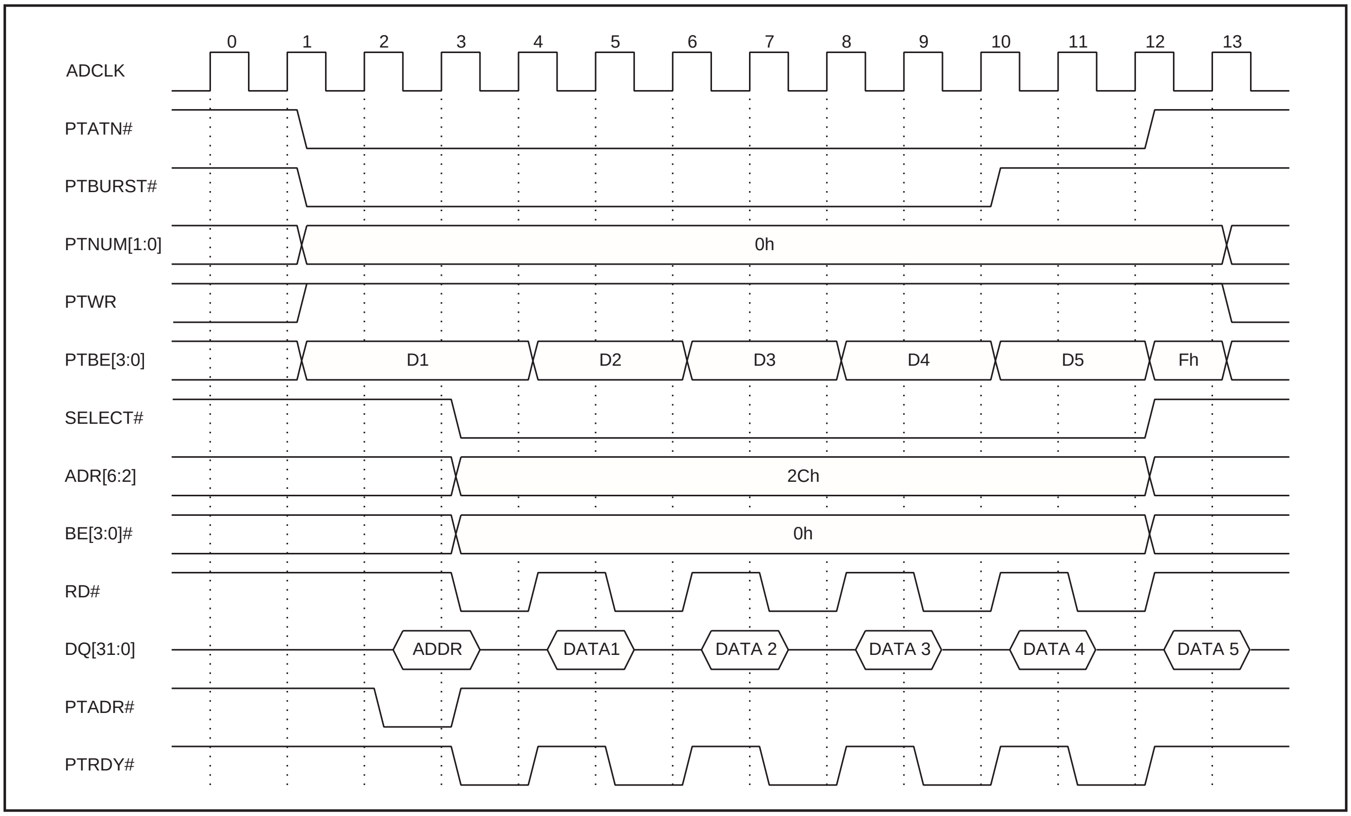
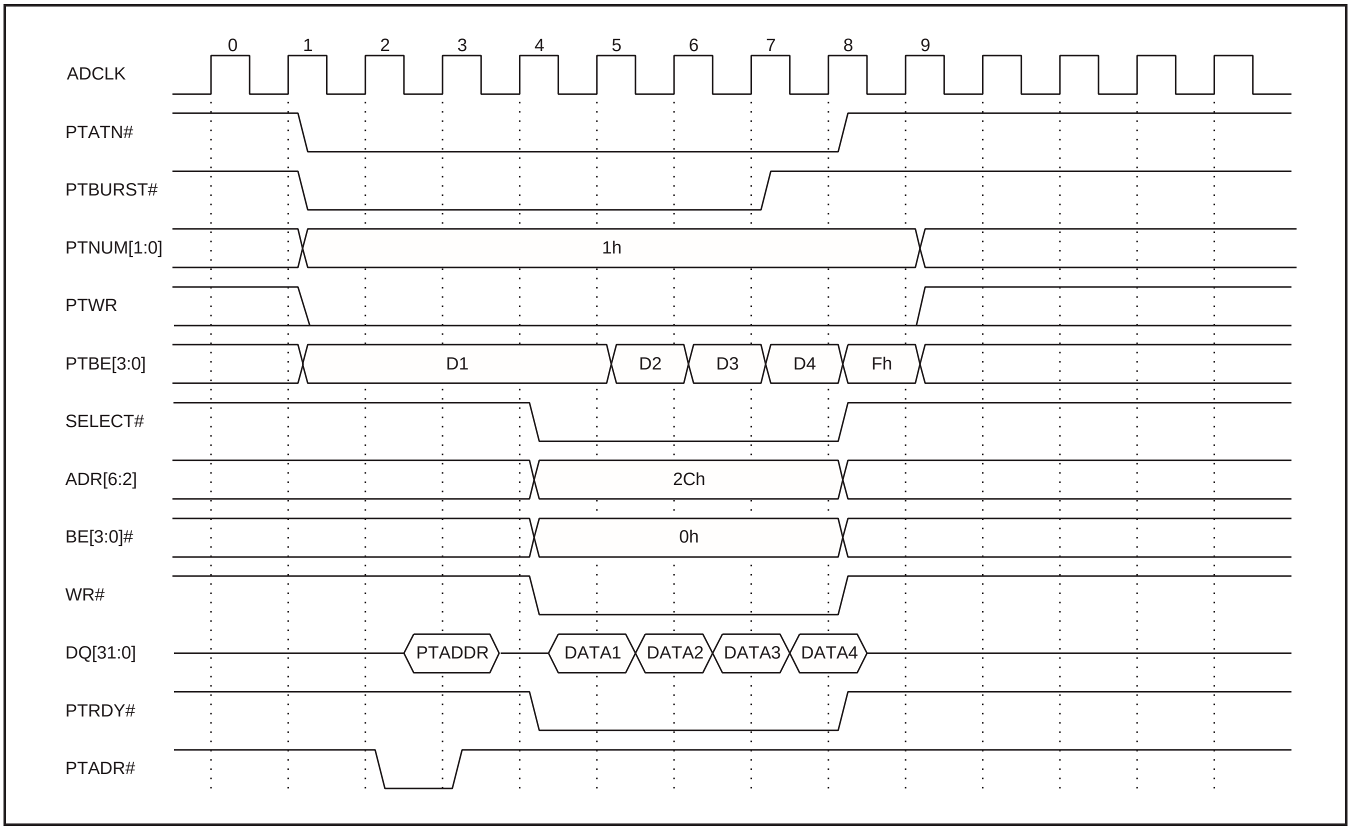


Figure 9. PCI to Add-On Passive Burst Read Access



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

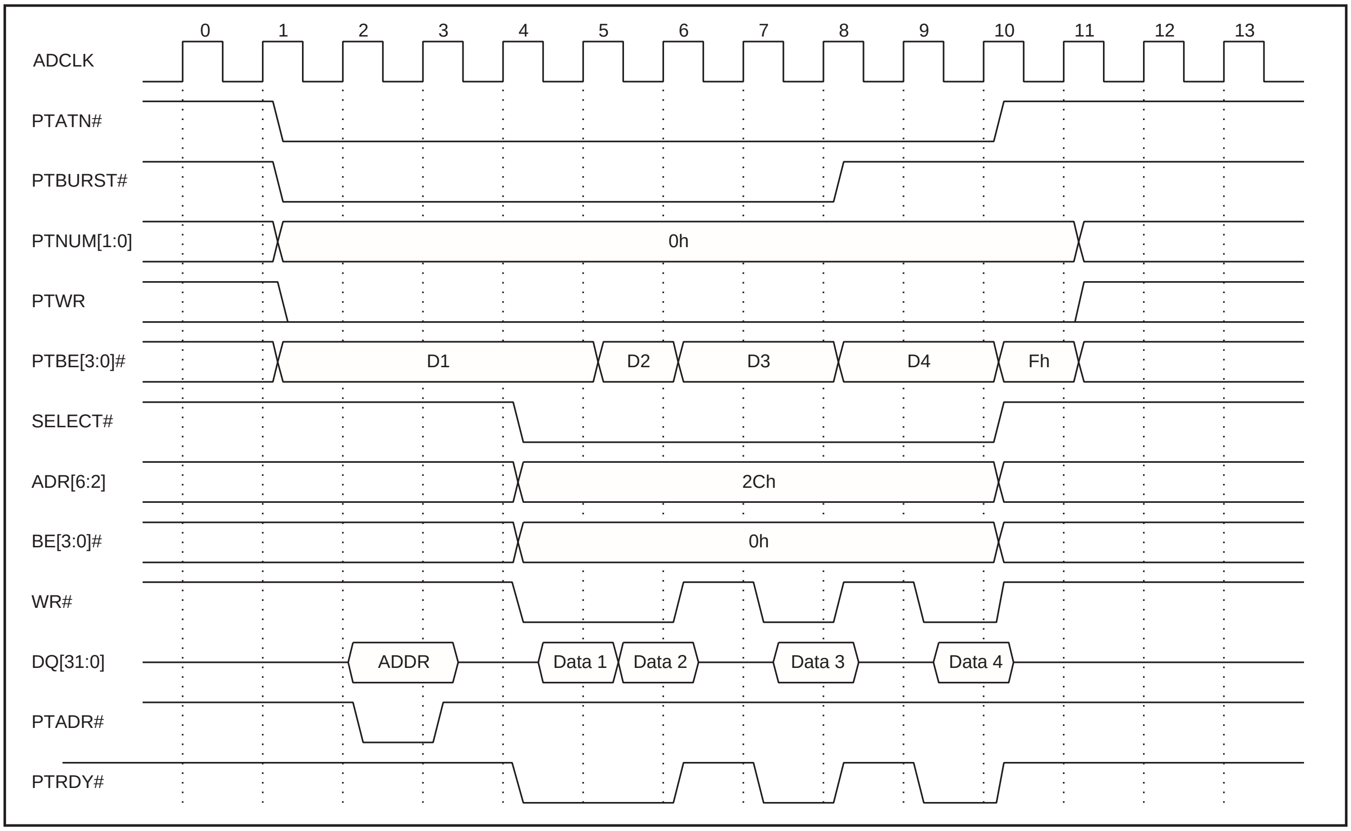


Figure 9 shows a Passive Mode Pass-Thru burst read access (Add-On write) of four DWORDs, using PTADR# to provide an address-phase.

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#.

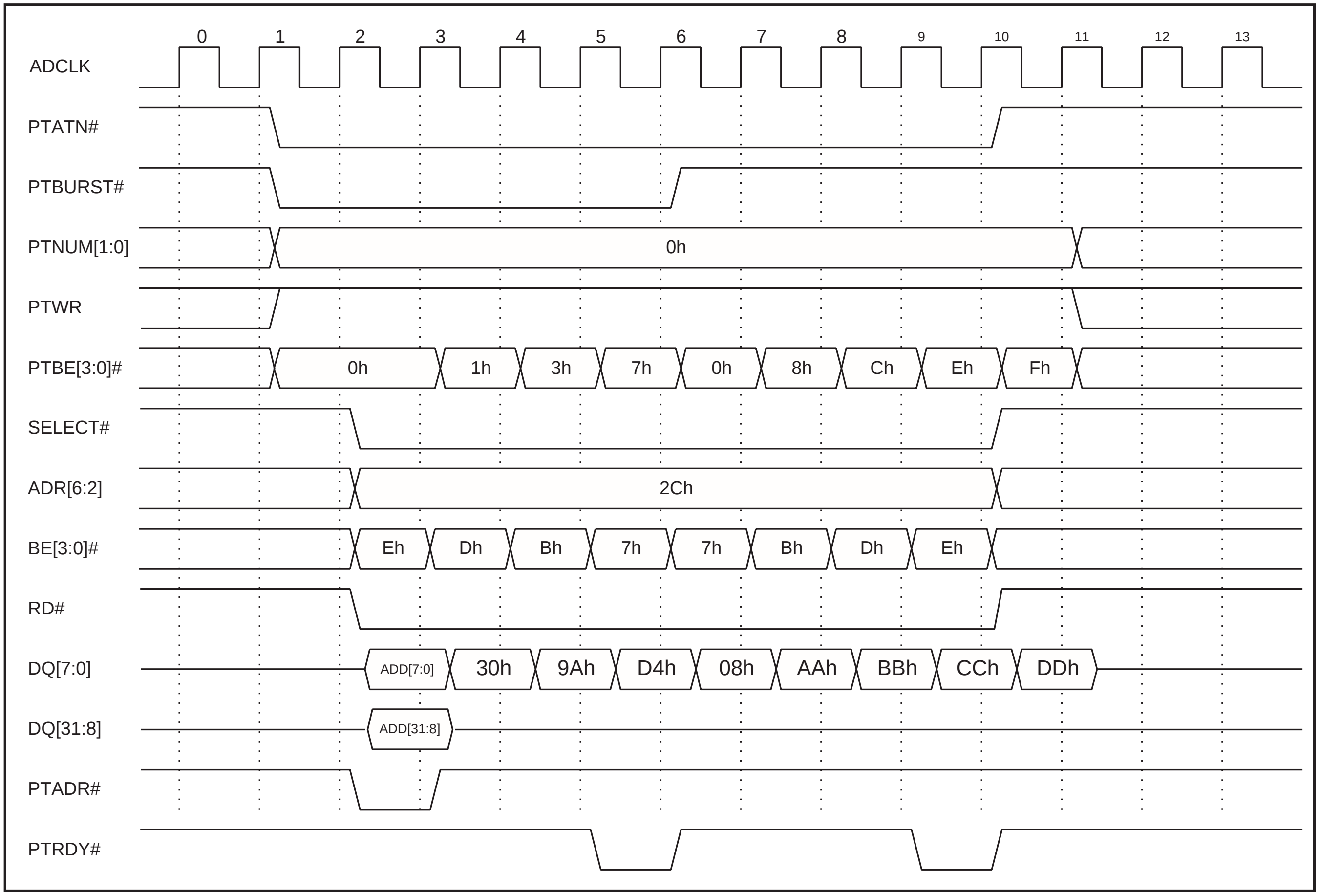
PTATN#	Asserted. Indicates Pass-Thru access is pending.
PTBURST#	Asserted. The access has multiple data phases.
PTNUM[1:0]	0h. Indicates the access is to Pass-Thru region 0.
PTWR	Asserted. Indicates the access is a write.
PTBE[3:0]#	0h. Indicates valid bytes for the first data transfer.

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

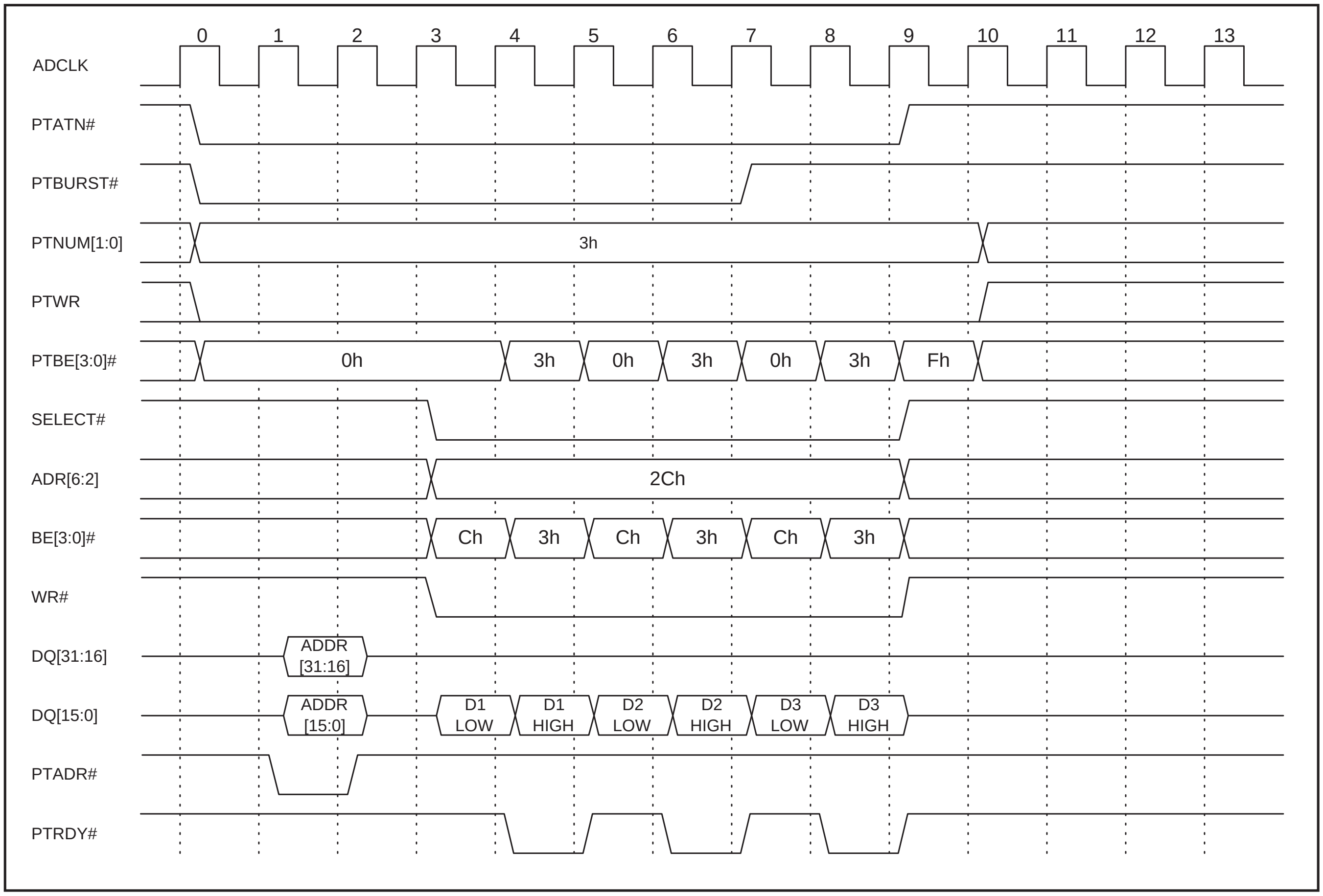


Table 4. Big Endian conversion for a 16-bit bus. The S5920 drives D[15:0] only

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#

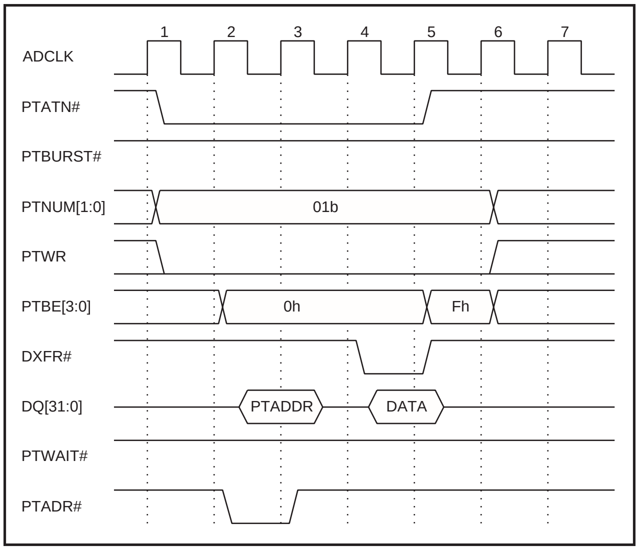
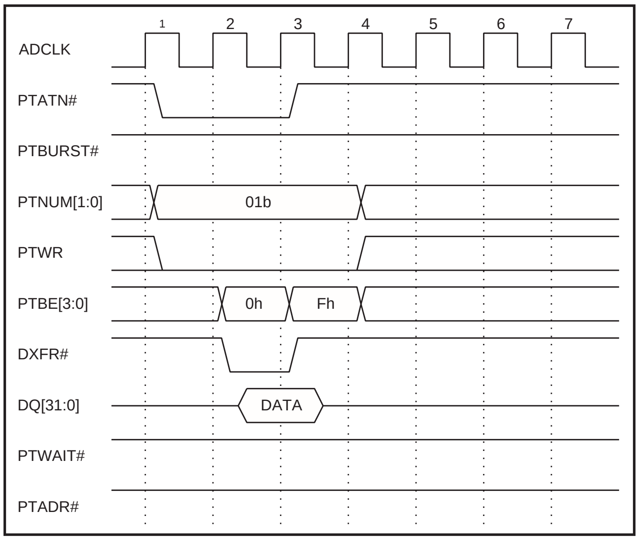


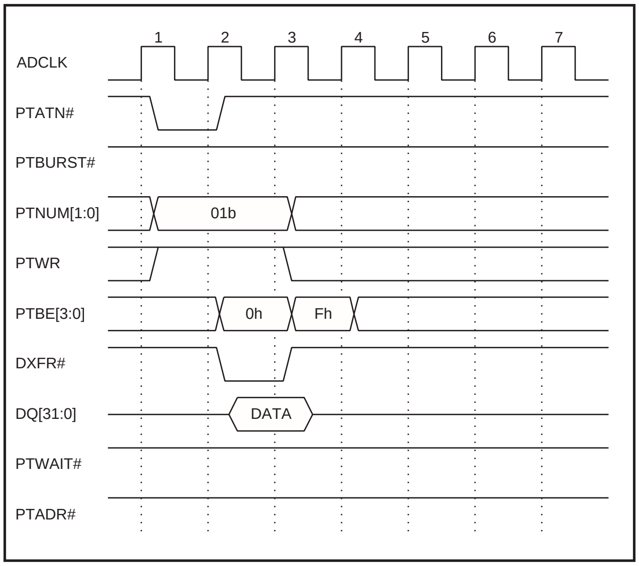
Figure 13b. Active Mode PCI Read without PTADR#



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#

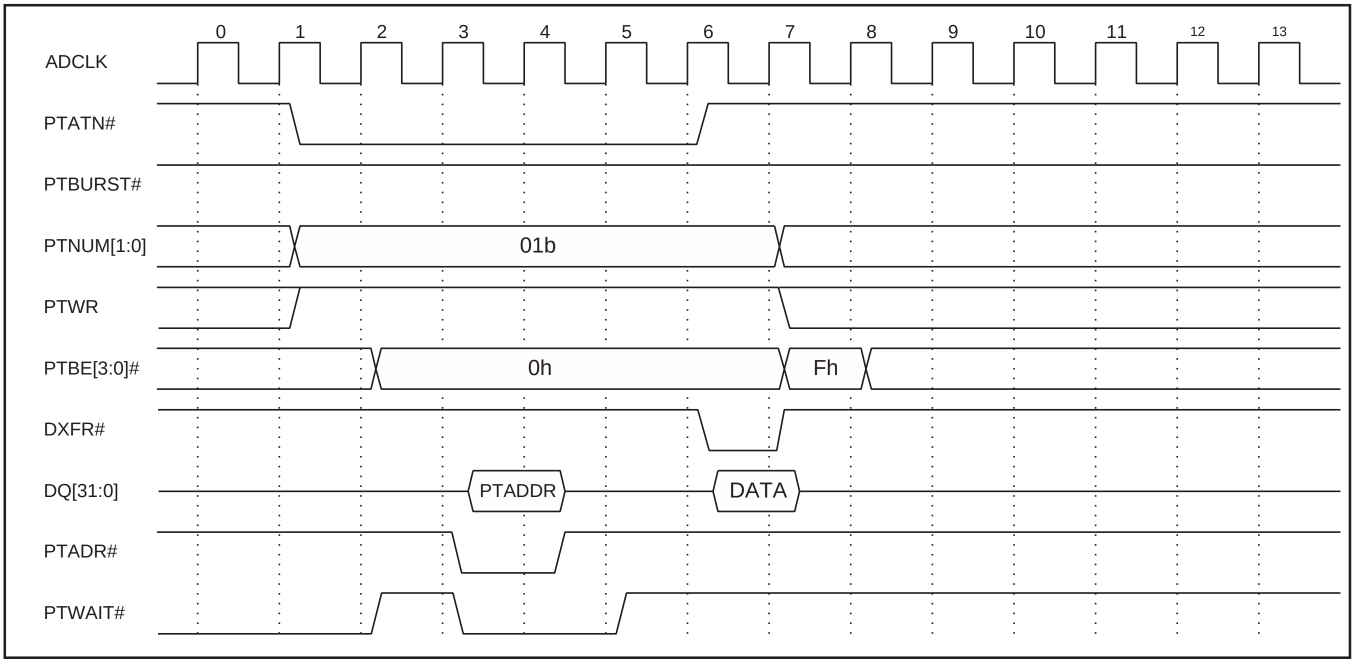
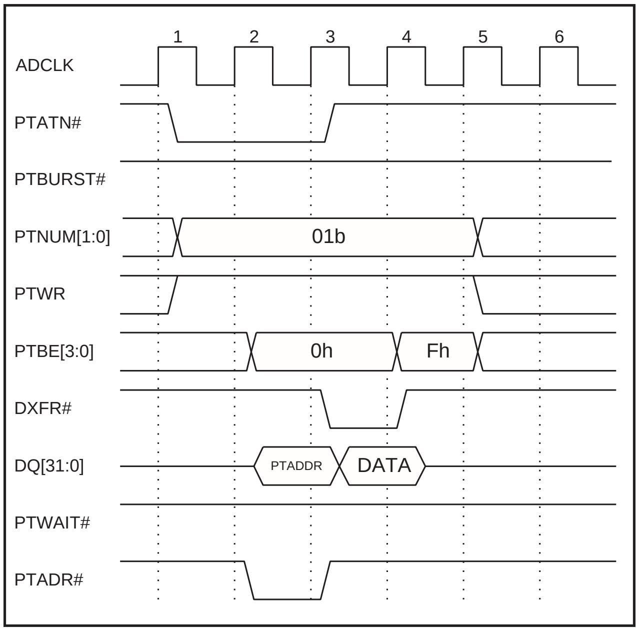


Figure 15. Active Mode 32-Bit PCI Write



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.

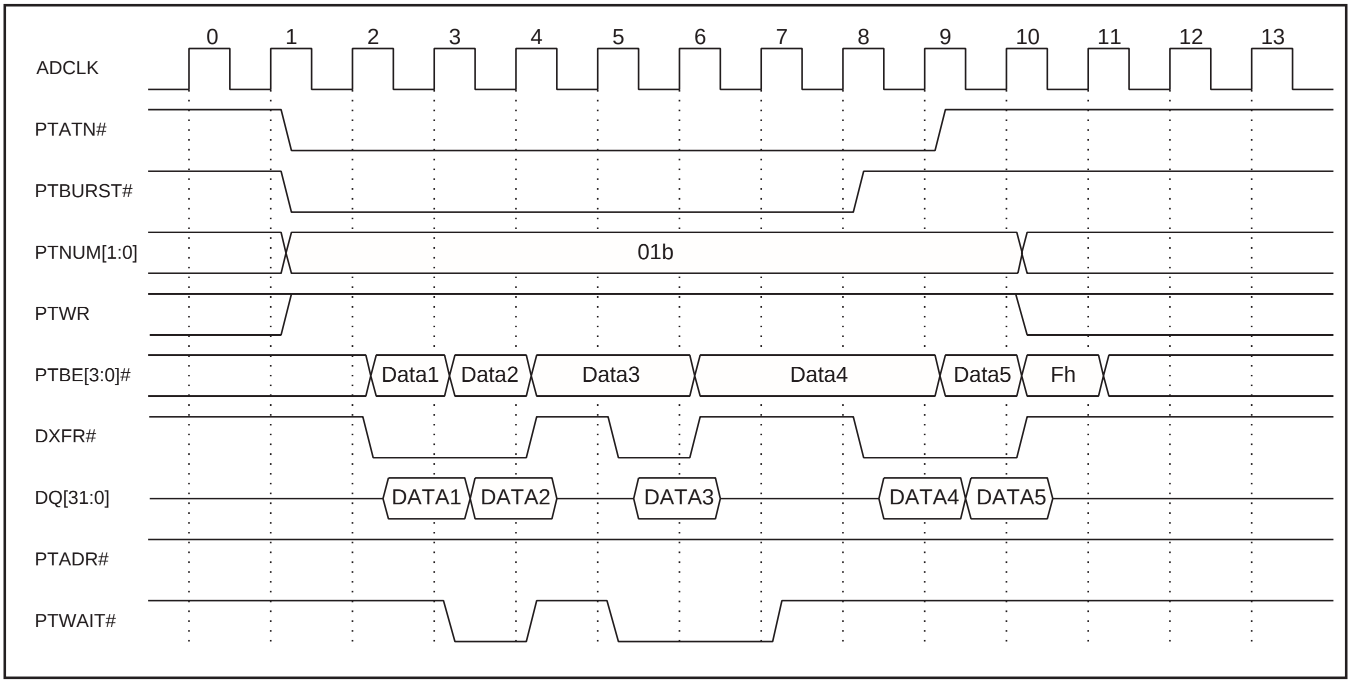
Figure 15 shows a single data phase 32-bit Active mode PCI Write with PTADR# enabled.

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

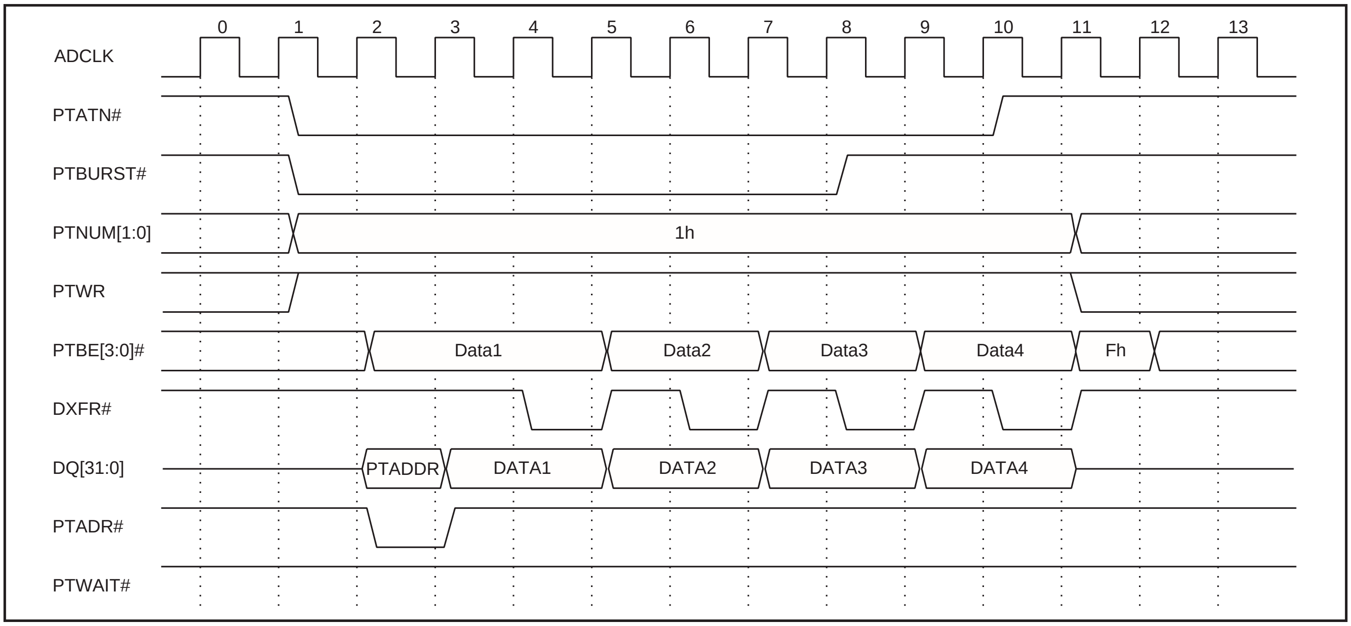
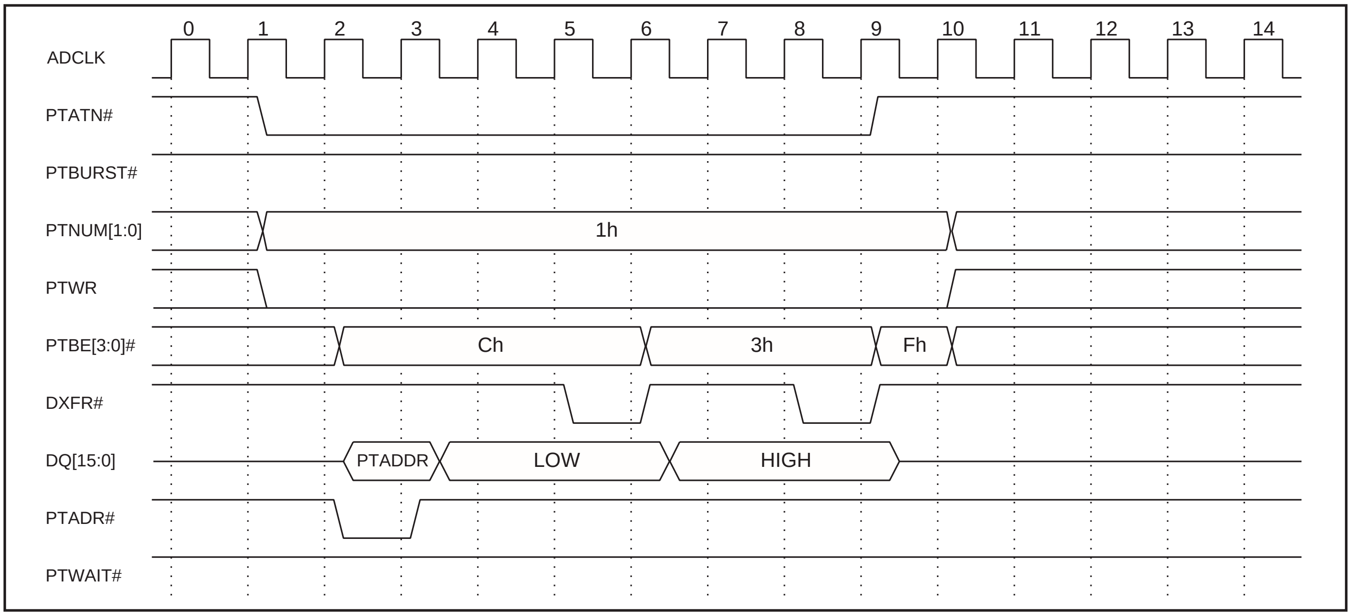


Figure 18. 16-Bit Active Mode PCI Read w/ Programmed Wait States



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

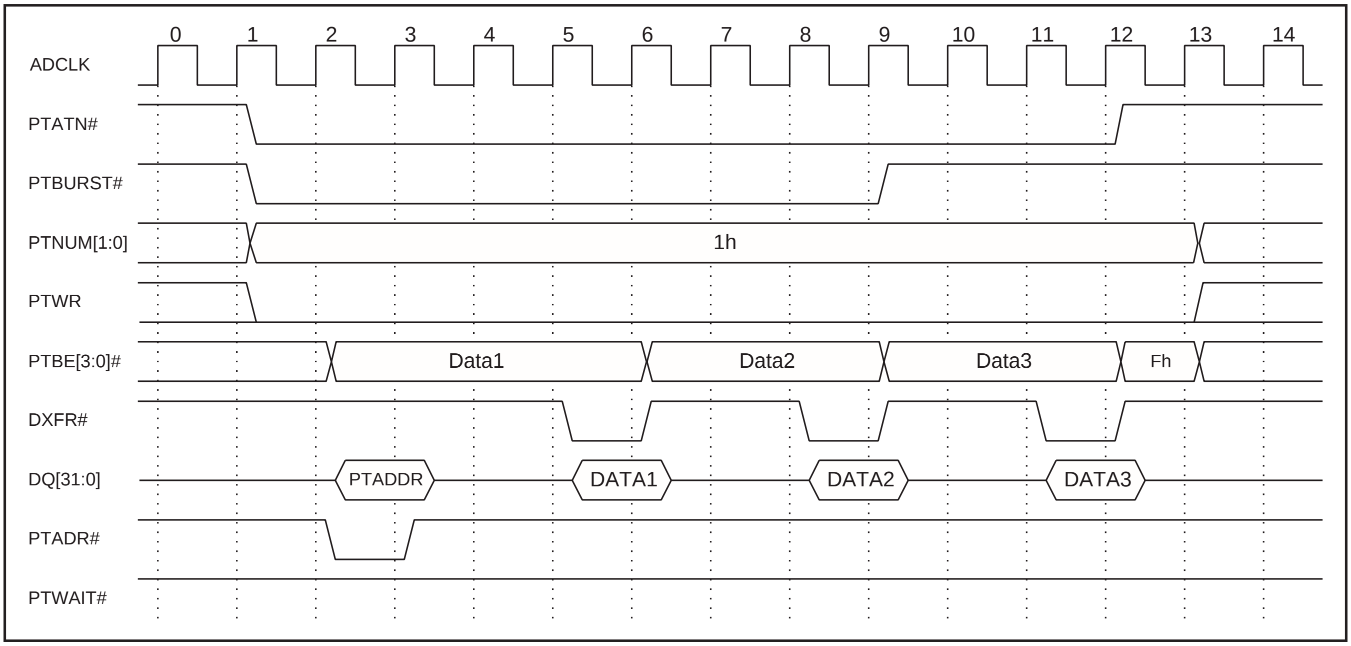


Figure 20. Active Mode PCI Read

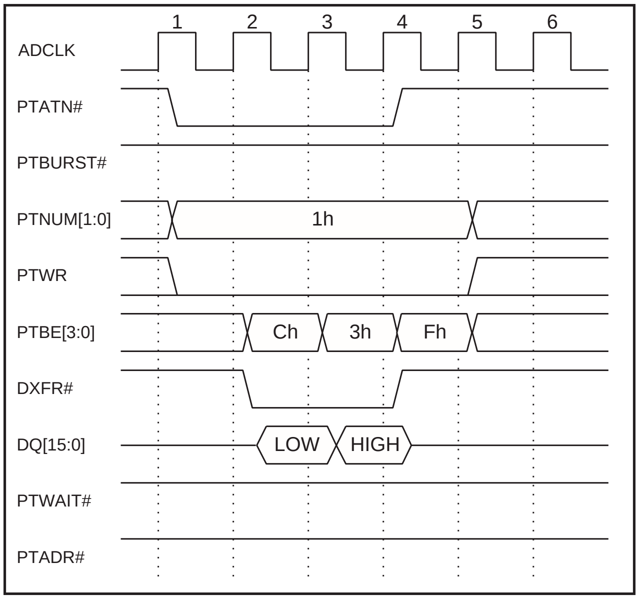


Figure 21. Active Mode PCI Write

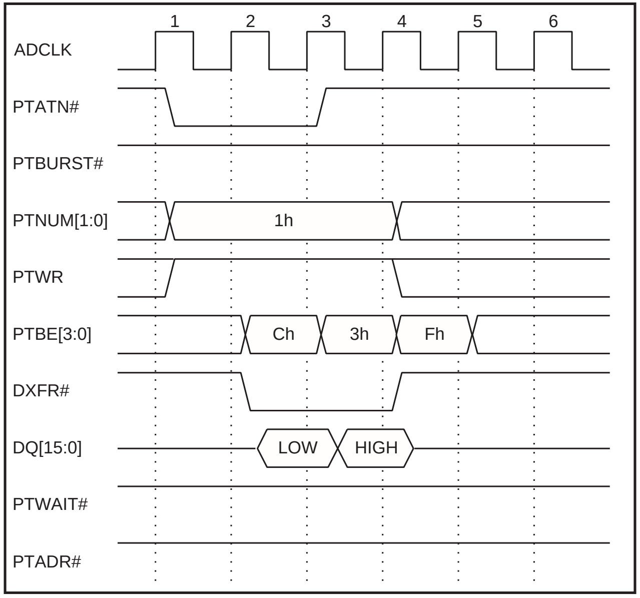


Figure 20 shows a Pass-Thru write cycle with 0 wait states for a 16-bit region. The PTBE[3:0]# signals are used by the Add-On device to determine validity of the current word cycle and also, which word of a long word is currently being driven by the S5920. PTBE[3:0]# encoding of Ch indicates the least 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.

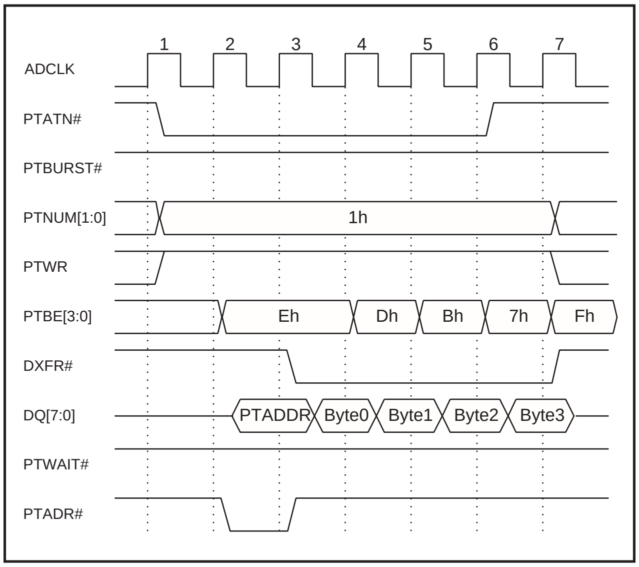
D31:30 Pass-Thru region bus width. These two bits are used by the S5920 to define the data bus width for a Pass-Thru region. Regardless of the programming of other bits in the BADR register, if D31:30 are zeros, the Pass-Thru region is disabled.

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%, 0^0C 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		
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

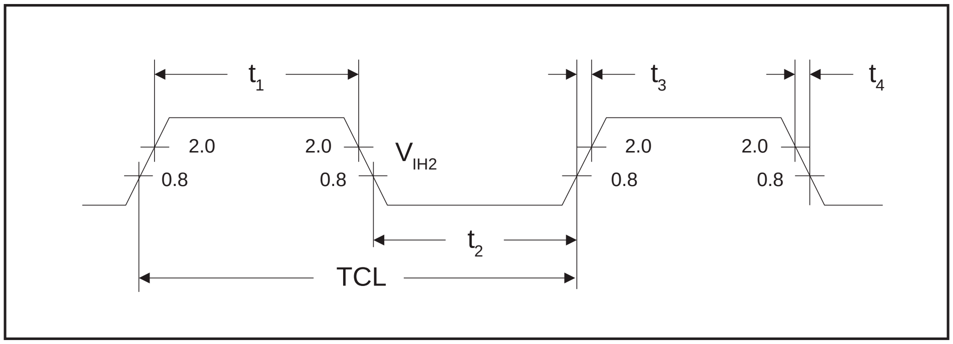
Functional Operation Range (V_{CC} = 5.0V +/- 5%, 0°C to 70°C, 50 pF load on outputs for MAX, 0 pF load for MIN).

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

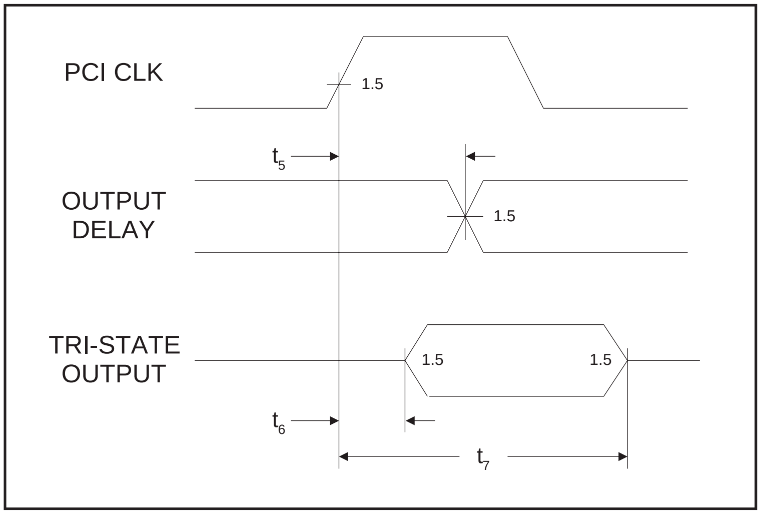
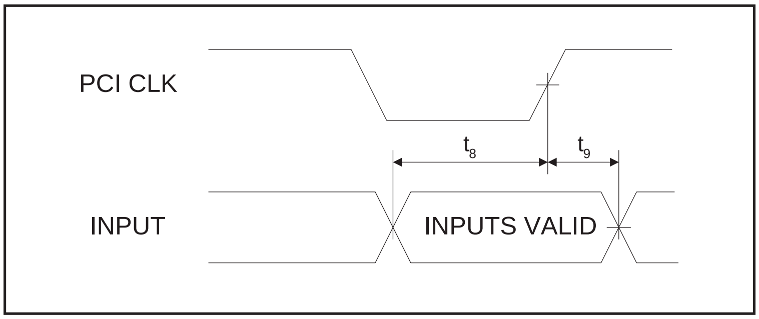


Figure 3. PCI Signal Input Timing



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%, 0^{o}C to $7^{o}0$ 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

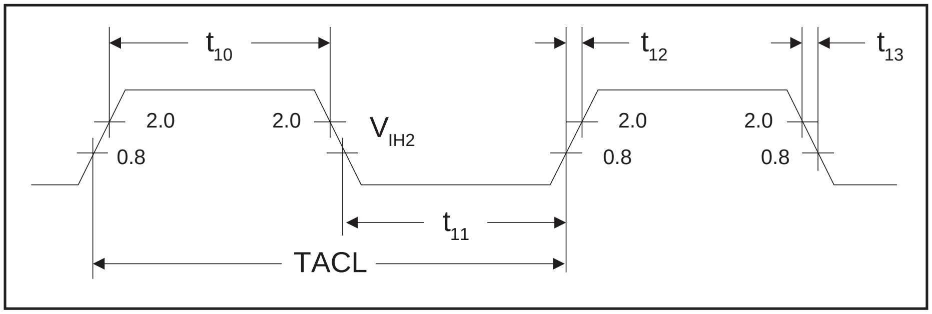


Figure 5. Pass-Thru Clock Relationship to PCI Clock

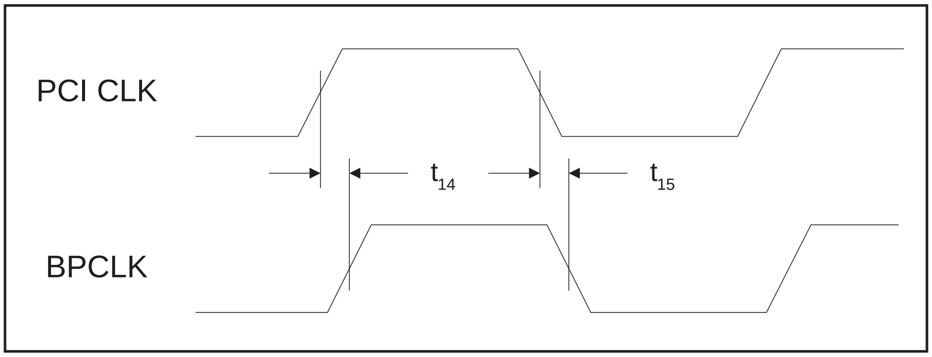


Figure 6. PTADR Timing

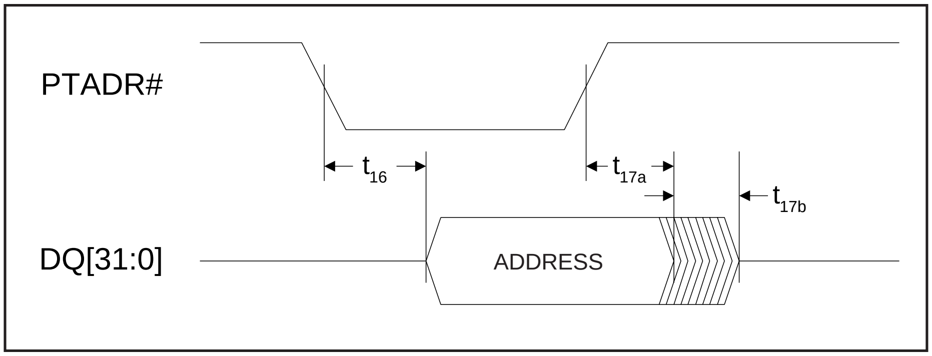


Figure 7. Passive Mode Pass-Thru Operation

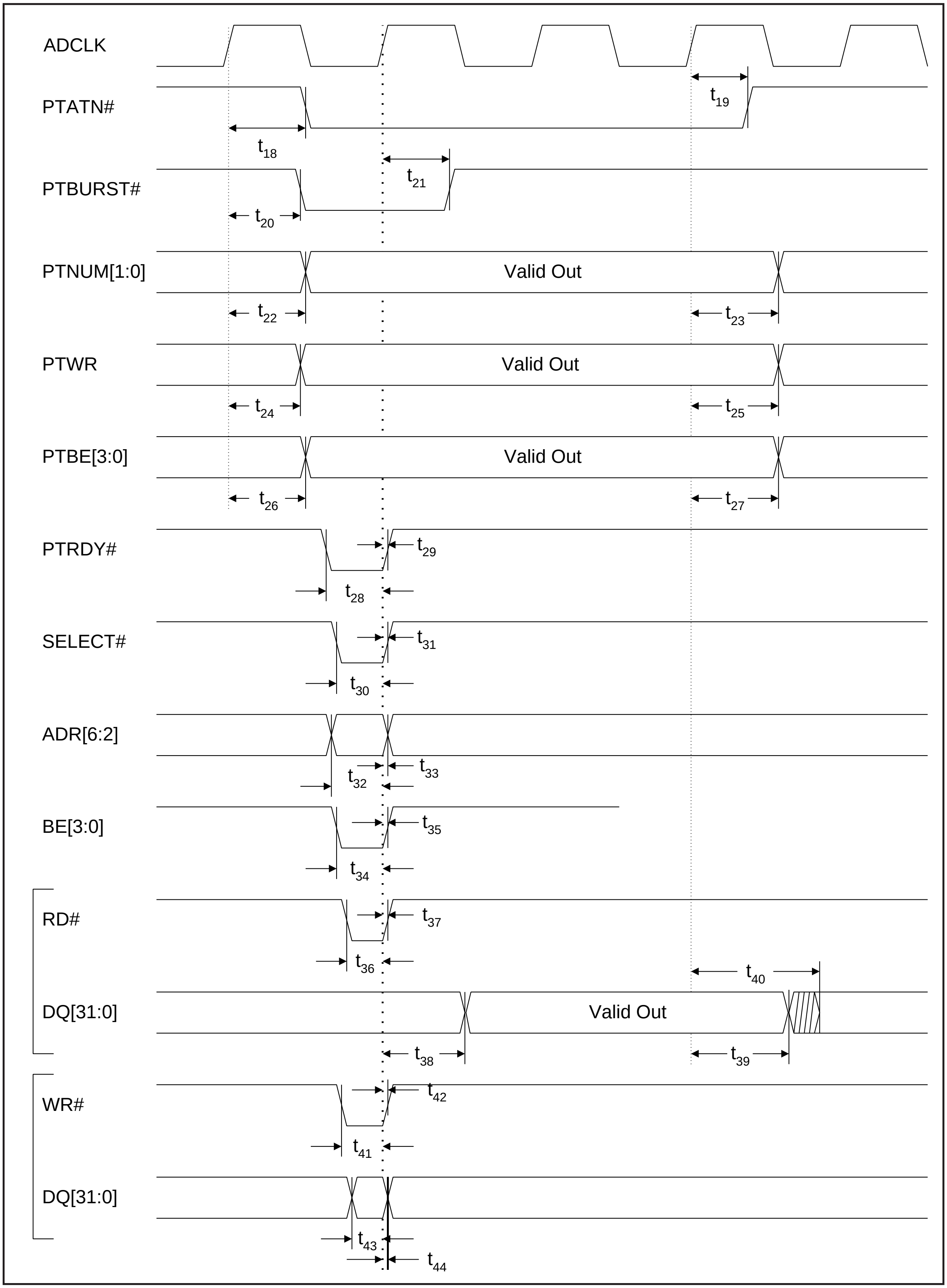


Table 7. Add-On Timings (Continued)

Functional Operation Range (V_{CC} = 5.0V +/- 5%, 0^{O}C to 70^{O}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

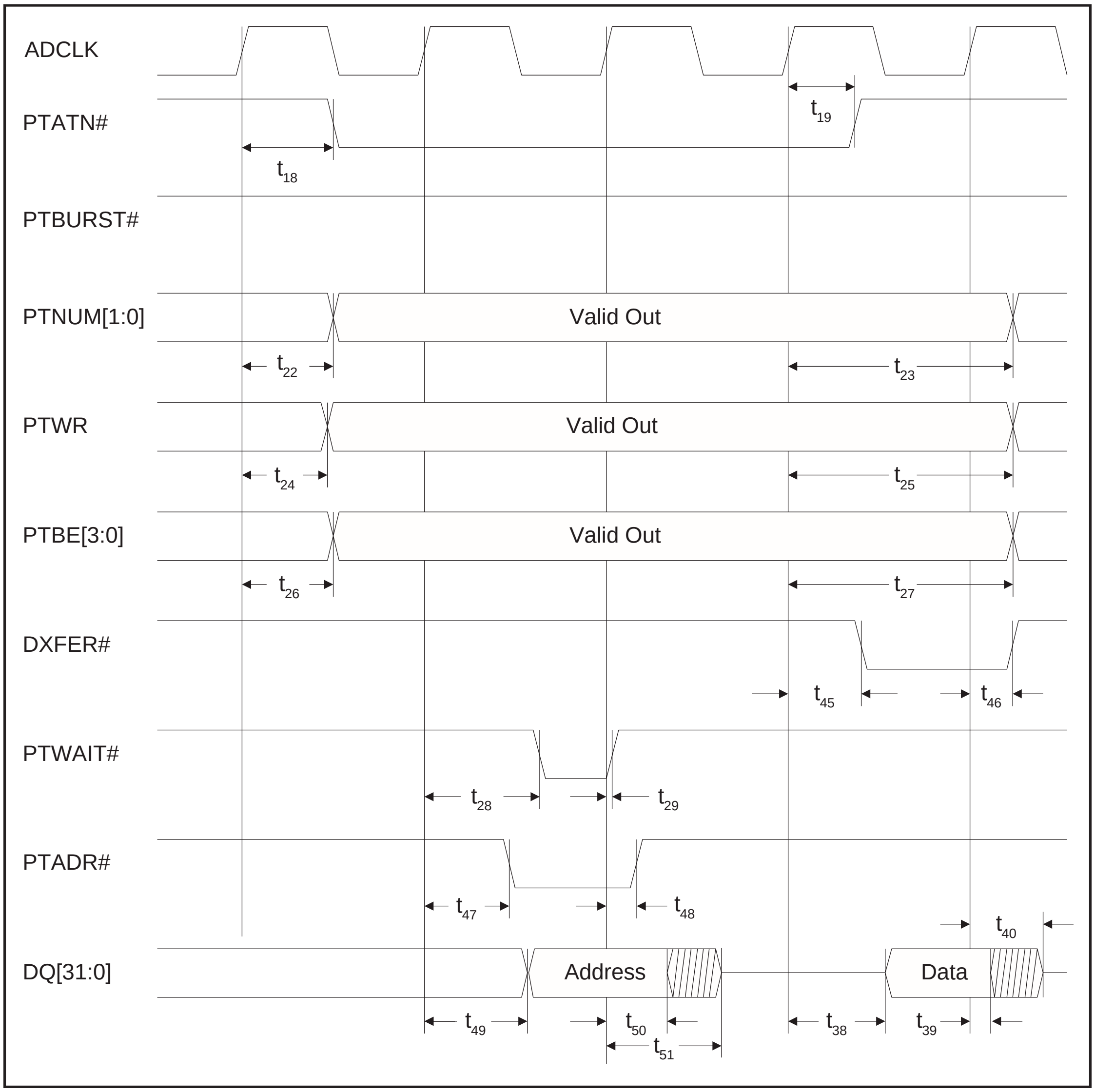


Table 8. Mailbox Timings

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

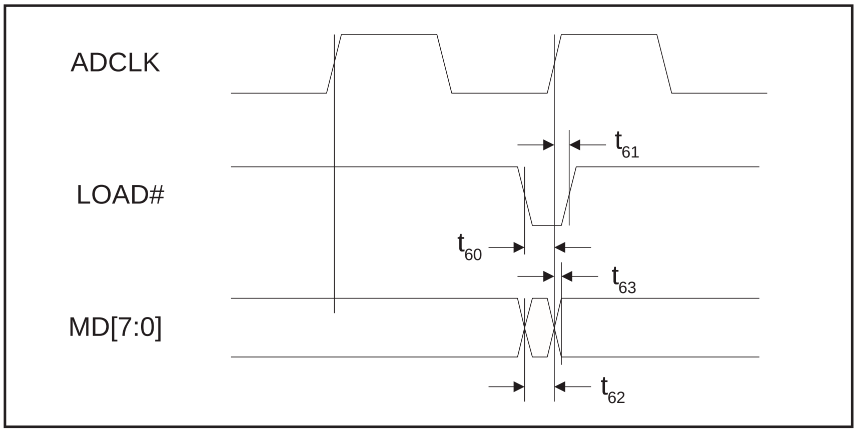


Figure 10. Mailbox Data

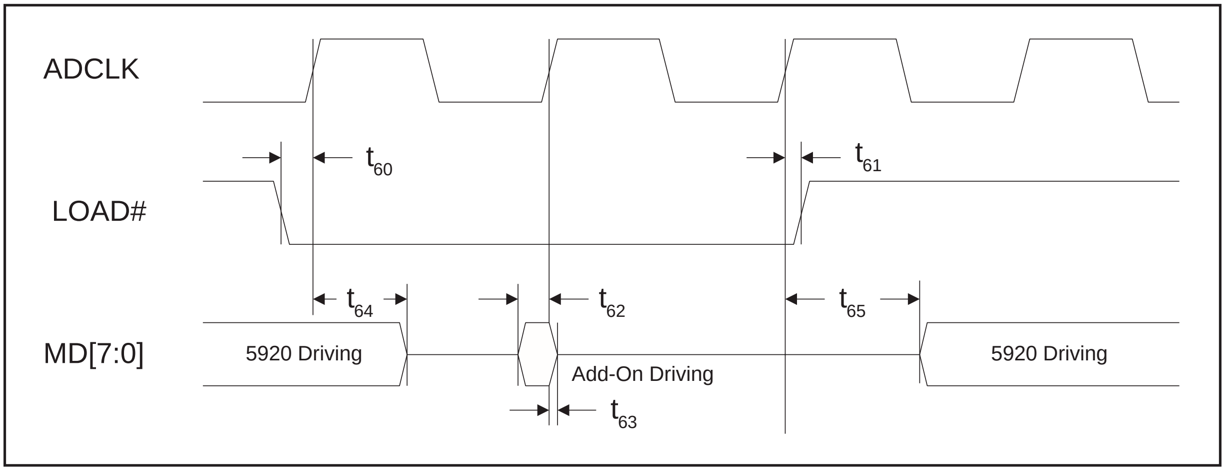


Figure 1. S5920 Pinout and Pin Assignment

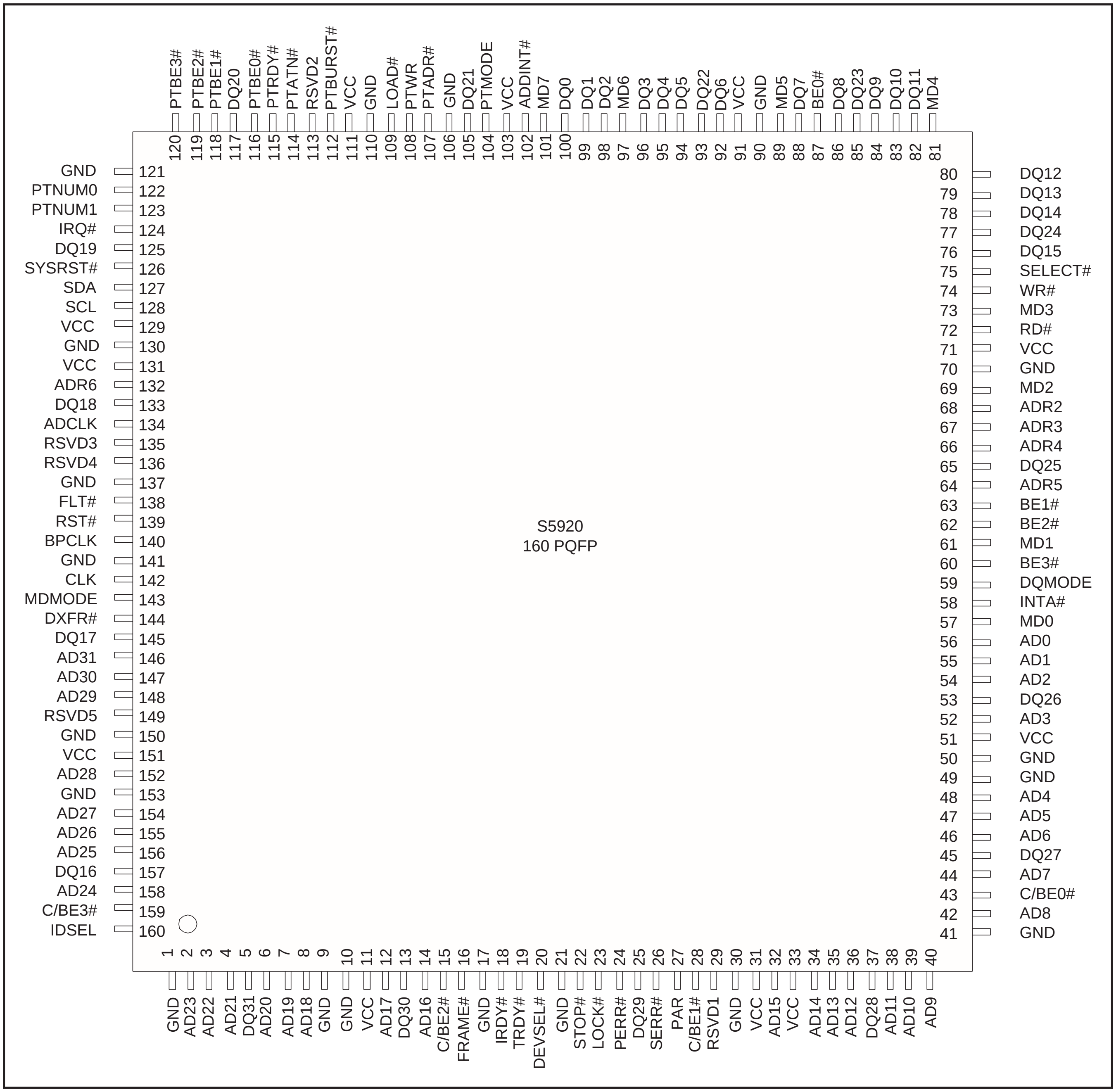
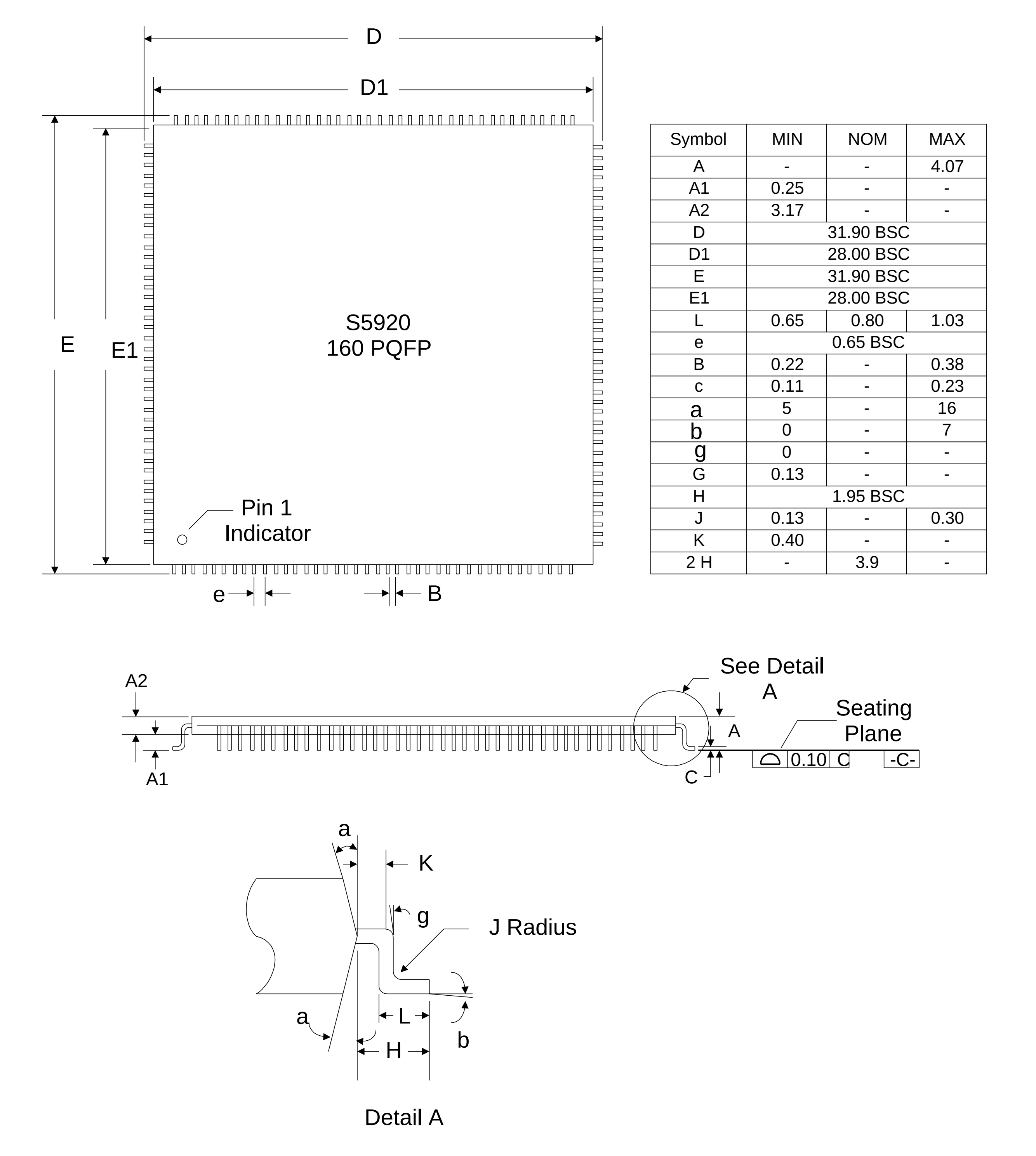
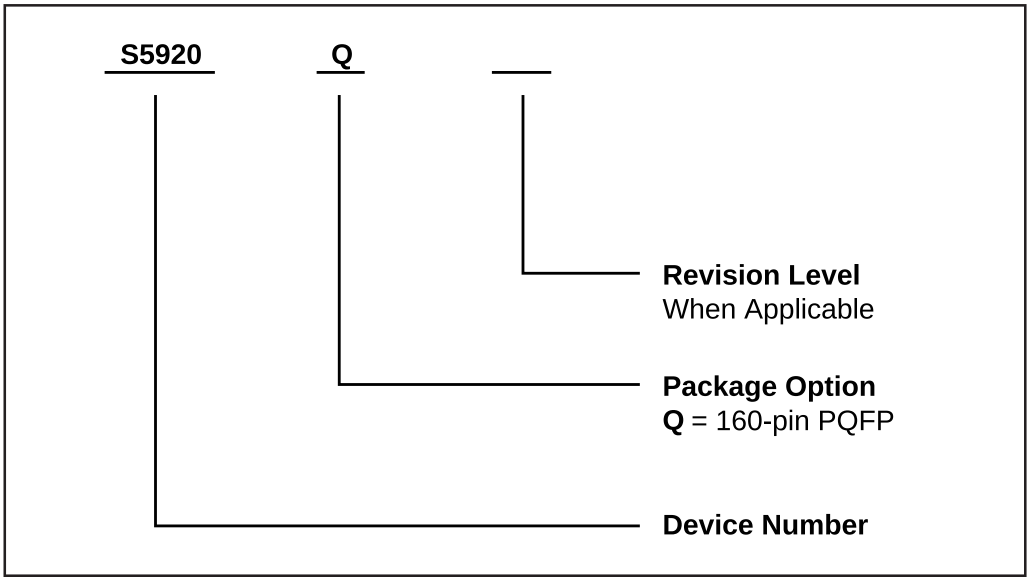


Figure 3. 160 PQFP (28 x 28 x 3.37 mm) - Plastic Quad Flat Package



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	-

Figure 4. Ordering Information



INDEX