TEXAS INSTRUMENTS

PCI1131 CardBus Controller Interrupts

Application Report

PCI bus

1997

Texas Instruments

Printed in U.S.A., 1197

SCPA007

TEXAS INSTRUMENTS

PCI1131 CardBus Controller Interrupts

PCI bus

1997

Printed in U.S.A., 1197

SCPA007

Application Report

[redacted]

[redacted]

PCI1131 CardBus Controller Interrupts

SCPA007
November 1997

PRINTED WITH SOY INK™

Printed on Recycled Paper

Contents

List of Figures

PCI1131 CardBus Controller Interrupts

ABSTRACT

Interrupts are an integral component in any computer architecture. Because of the dynamic nature of the Personal Computer Memory Card International Association (PCMCIA) subsystem and the abundance of PC Card™ input/output (I/O) applications, interrupts are also an integral part of the PCI1131 CardBus controller. The PCI1131 provides several interrupt signaling schemes to accommodate the needs of a variety of platforms. The different mechanisms for dealing with interrupts, either parallel ISA, parallel PCI, or serialized ISA, are reviewed in reference to the PCI1131 CardBus controller.

1 Introduction

The different mechanisms for dealing with interrupts in the PCI1131 are based on various specifications and industry standards. The ExCA register set provides interrupt control for some 16-bit PC Card functions, and the *PCMCIA CardBus Specification* [1] provides interrupt control for the CardBus PC Card functions. Therefore, the PCI1131 is backward compatible with existing register definitions and defines new registers as required.

2 Interrupts

The PCI1131 detects interrupts and/or events at the PC Card interface and notifies the host interrupt controller through one of several interrupt signaling protocols. To simplify the discussion and use of interrupts in the PCI1131, PC Card interrupts are classified as either card-status-change (CSC) interrupts or functional interrupts. Functional interrupts are explicit requests for interrupt servicing directly from the PC Card application. Such requests are communicated over a dedicated PC Card signal defined for this purpose. CSC interrupts indicate a change in the state of the PC Card (for example, card removal or insertion, or power-up complete). All sources of functional and CSC interrupts are discussed in detail, as well as any specific options to be configured by the host software.

The method by which either type of PC Card interrupt is communicated to the host interrupt controller varies from system to system. The PCI1131 offers system designers the choice of using either peripheral component interconnect (PCI) interrupt signaling with traditional industry standard architecture (ISA) interrupt request (IRQ) signaling or the serialized ISA IRQ protocol (see Figure 1).

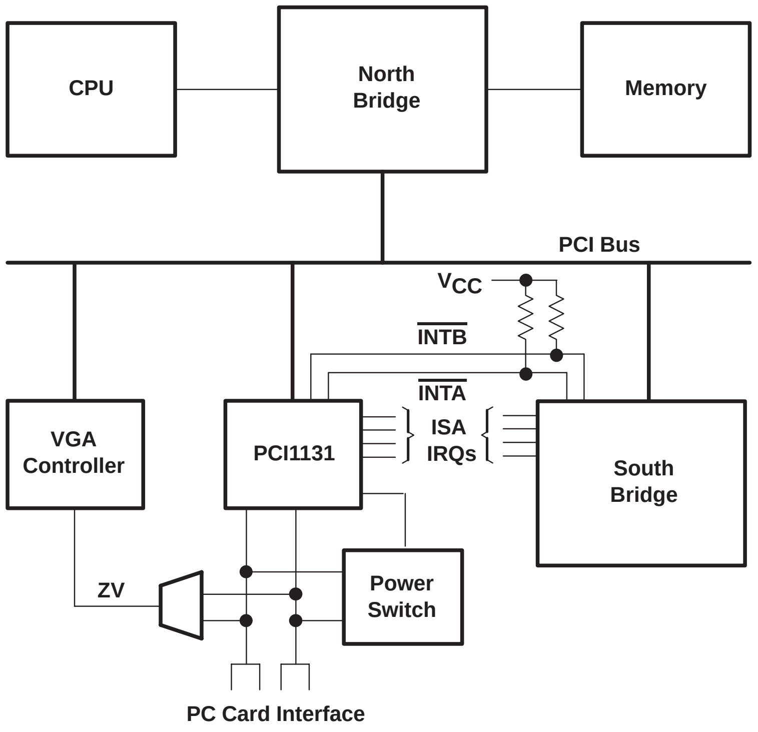

Figure 1. System Using Parallel ISA IRQ and PCI Interrupts

In Figure 1, the PCI1131 is connected to the interrupt controller of the chipset south bridge with parallel ISA IRQs and parallel PCI interrupts. In this case, the PCI1131 must be configured at boot time by the basic input/output system (BIOS) to support PCI interrupts and to select the ISA IRQs in the PCI configuration register. When using Win95 OSR2, the operating system uses the PCI interrupts for CSC events, such as card insertion and removal. The functional interrupts are routed according to the type of PC Card that is inserted. If the PC Card is a CardBus card, the system must use the PCI interrupts to signal the functional card interrupts. If the PC Card is a 16-bit I/O PC Card, the functional card interrupts are signaled on one of the ISA IRQ lines and the particular ISA IRQ is selected in the socket's ExCA registers.

In Figure 2 and Figure 3, the systems also use parallel PCI interrupts; however, they output the ISA IRQs on the IRQSER pin of the PCI1131. This serial interrupt scheme is fully compliant with the *Serialized IRQ Support for PCI Systems*, *Revision 6.0*, September 1, 1995 release [2]. While the PCI1131 is fully compliant with the serialization specification, it does not support the PCI interrupts on the IRQSER stream, which are included in the specification. In the system shown in Figure 2, the PCI1131 is directly connected to the programmable interrupt controller (PIC) with the IRQSER. This is done with chipsets that support this serial interrupt mode.

Use a strong pullup resistor of approximately 1 kΩ on the IRQSER line when implementing serial interrupts. If the serialized interrupt mode is not supported by the chipset and you want to use the IRQSER output from the PCI1131, you can use the PCI950 deserializer chip to deserialize the IRQSER output.

Details for implementing each of these possible interrupt schemes are given in the following paragraphs.

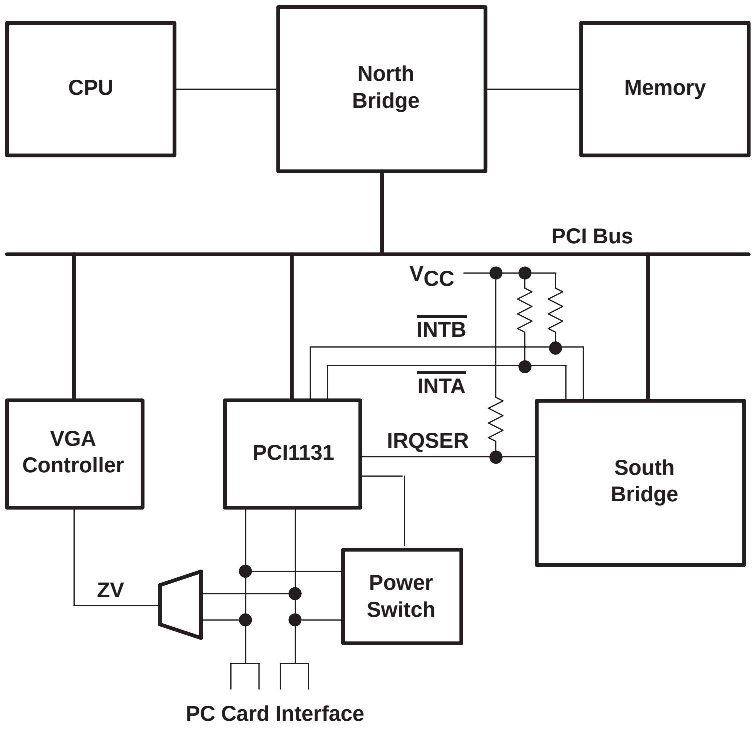

Figure 2. System Using Serial ISA IRQ and Parallel PCI Interrupts

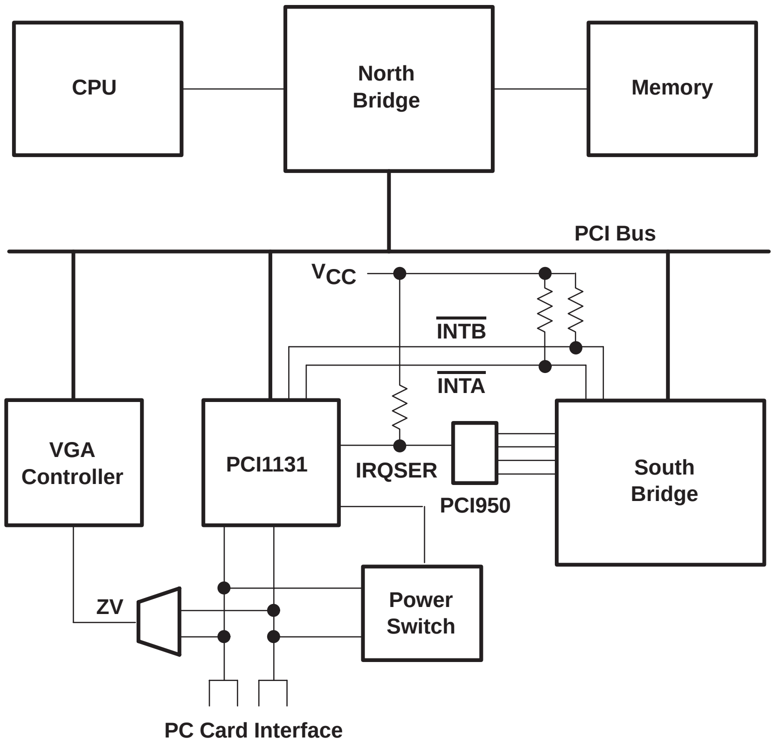

Figure 3. System Using Serial ISA IRQs That Are Deserialized by the PCI950 Deserializer and Parallel PCI Interrupts

2.1 Functional and CSC Interrupts

Functional interrupts are requests from a PC Card application for interrupt service, and are indicated by asserting specially defined signals on the PC Card interface. Functional interrupts are generated by 16-bit I/O PC Cards and by CardBus cards. CSC interrupts are defined as events at the PC Card interface, which are detected by the PCI1131 and may warrant notification of host software for service. Such events include transitions on certain PC Card signals or card removal or insertion. The specific examples of functional and CSC interrupts depend on the type of PC Card(s) installed in the socket at any given time. A CardBus card has entirely different methods than a 16-bit card for signaling interrupts, and the 16-bit interrupt sources differ even between memory and I/O PC Cards.

Table 1 summarizes the sources of interrupts and the types of PC Cards associated with them. Functional interrupt events are valid only for 16-bit I/O and CardBus PC Cards. Also, card insertion and removal events are independent of the card type. This is because the same signals are used in both cases; the card-detect signals and the PCI1131 cannot distinguish between card types at the time of card insertion. An interrogation process is used to determine the card type, that is, 16-bit or CardBus.

Table 1. PC Card Interrupt Events and Description

CARD TYPE	EVENT	TYPE	SIGNAL	DESCRIPTION
16-bit memory	Battery conditions (BVD1, BVD2)	CSC	BVD1($\overline{\text{STSCHG}}$)//CSTSCHG	A transition on BVD1 indicates a change in the PC Card battery conditions.
		CSC	BVD2($\overline{\text{SPKR}}$)//CAUDIO	A transition on BVD2 indicates a change in the PC Card battery conditions.
	Wait states (READY)	CSC	READY($\overline{\text{IREQ}}$)//$\overline{\text{CINT}}$	A transition on READY indicates a change in the ability of the memory PC Card to accept or provide data.
16-bit I/O	Change in card status ($\overline{\text{STSCHG}}$)	CSC	BVD1($\overline{\text{STSCHG}}$)//CSTSCHG	Assertion of $\overline{\text{STSCHG}}$ indicates a status change on the PC Card.
	Interrupt request ($\overline{\text{IREQ}}$)	Functional	READY($\overline{\text{IREQ}}$)//$\overline{\text{CINT}}$	Assertion of $\overline{\text{IREQ}}$ indicates an interrupt request from the PC Card.
CardBus	Change in card status (CSTSCHG)	CSC	BVD1($\overline{\text{STSCHG}}$)//CSTSCHG	Assertion of CSTSCHG indicates a status change on the PC Card.
	Interrupt request (CINT)	Functional	READY(IREQ)//CINT	Assertion of $\overline{\text{CINT}}$ indicates an interrupt request from the PC Card.
	Power cycle complete	CSC	N/A	An interrupt is generated when a PC Card power-up cycle has completed.
All PC Cards	Card insertion or removal	CSC	$\overline{\text{CD1}}$//$\overline{\text{CCD1}}$, $\overline{\text{CD2}}$//$\overline{\text{CCD2}}$	A transition on either $\overline{\text{CD1}}$//$\overline{\text{CCD1}}$ or $\overline{\text{CD2}}$//$\overline{\text{CCD2}}$ indicates an insertion or removal of a 16-bit//CardBus PC Card.
	Power cycle complete	CSC	N/A	An interrupt is generated when a PC Card power-up or power-down cycle has completed.

The signal-naming convention for PC Card signals describes the function for 16-bit memory and I/O cards, as well as CardBus. The 16-bit memory card signal name is first, with the I/O card signal name second, enclosed in parentheses. The CardBus signal name follows after a double forward slash (//). The 16-bit I/O and CardBus PC Cards both have similar methods for signaling interrupts. Both use two signals: one to indicate a change in card status and another to request interrupt service from the host. A 16-bit memory PC Card uses the BVD1 and BVD2 signals to indicate changes in battery conditions on the card and the READY signal to insert wait states during memory-card data transfers.

The PC Card standard describes the power-up sequence that must be followed by the PCI1131 during an insertion event and when the host requests that the socket V_{CC} and V_{PP} be powered. Upon completion of this power-up sequence, the PCI1131 interrupt scheme may be used to notify the host system, as indicated in Table 1, denoted by *power cycle complete*. This interrupt source is considered a PCI1131 internal event because it does not depend on a signal change at the PC Card interface, but rather the completion of applying power to the socket.

By setting the appropriate bits in the PCI1131, the host software can individually mask (disable) each of the potential CSC interrupt sources listed in Table 1. By individually masking the interrupt sources listed in Table 1, the host software can control the events that cause a PCI1131 interrupt. By programming the appropriate routing registers, the host software can partially control which system interrupt the PCI1131 is going to assert . The PCI1131 host software routes PC Card CSC and functional interrupts to separate system interrupts. Interrupt routing is somewhat specific to the interrupt signaling method used and is discussed in more detail later.

When an interrupt is signaled by the PCI1131, the interrupt service routine must be able to discern which of the events in Table 1 caused the interrupt. This is of particular interest with CSC interrupts where a variety of events at the card interface may cause interrupts. Internal registers in the PCI1131 provide flags that report to the host interrupt service routine which of the interrupt sources caused an interrupt. By first reading these status bits, the interrupt service routine can determine what action to take.

Table 2 describes the valid PC Card interrupt events and further details the internal PCI1131 registers associated with masking and reporting them.

Table 2. PC Card Interrupt Mask and Flag Registers

CARD TYPE	EVENT	MASK	FLAG
16-bit memory	Battery conditions (BVD1, BVD2)	ExCA offset 05h/45h/805h Bits 1 and 0	ExCA offset 04h/44h/804h Bits 1 and 0
	Wait states (READY)	ExCA offset 05h/45h/805h Bit 2	ExCA offset 04h/44h/804h Bit 2
16-bit I/O	Change in card status ($\overline{\text{STSCHG}}$)	ExCA offset 05h/45h/805h Bit 0 Always enabled	ExCA offset 04h/44h/804h Bit 0
	Interrupt request ($\overline{\text{IREQ}}$) Power cycle complete	Always enabled	PCI configuration offset 91h Bit 0
CardBus	Change in card status (CSTSCHG)	Socket mask register Bit 0	Socket event register Bit 0
	Interrupt request (CINT)	Always enabled	Socket present state register Bit 6
	Power cycle complete	Socket mask register Bit 3	Socket event register Bit 3
All PC Cards	Card insertion or removal	Socket mask register Bits 2 and 1	Socket event register Bits 2 and 1 ExCA offset 04h/44h/804h Bit 3

There are various methods of clearing the interrupt flag bit. ExCA provides two methods to clear 16-bit PC Card-related interrupt flags. One is the explicit writing of 1 to the bit in question. The other is simply reading from the register. The selection is made through bit 2 in ExCA offset 1Eh/5Eh/81Eh.

There is a single exception to Table 2 when PCI interrupt signaling is used. The enable/disable bits for functional and CSC interrupts are in separate registers in PCI configuration register 91h, bits 4 and 3. Refer to Section 2.3, *PCI Interrupts*, for details.

2.2 ISA IRQ Interrupts

NOTE: Texas Instruments recommends using 43-kΩ pullup resistors for all unused inputs.

Among the PCI1131 interrupt signaling schemes is the traditional ISA IRQ signaling, popular in most x86 PCs. Dedicated terminals on the PCI1131 assert ten of the 15 ISA IRQs: IRQ3, IRQ4, IRQ5, IRQ7, IRQ9, IRQ10, IRQ11, IRQ12, IRQ14, and IRQ15. These IRQs represent the common interrupts used in PC Card applications and several free IRQs for CSC routing.

In a system using ISA IRQs, the host software (system BIOS) must first configure the PCI1131 to use ISA signaling by setting bits 2–1 of the PCI configuration register offset 92h to 01b. The ten IRQ terminals remain in the three-state condition until the ExCA CSC and functional interrupt routing registers are set to a valid state.

The step-by-step events that the host software must follow to successfully configure the PCI1131 for ISA IRQ signaling are listed in the following procedure. These steps assume that the system powered up and that the PCI reset ($\overline{\text{RSTIN}}$) pin is set high (deasserted). In cases where only selected bits of a register are to be modified, the host software leaves the remaining register bits unchanged first by reading the current contents of the register, then by modifying the desired bits, and last by writing the new value back to the respective PCI1131 registers.

The procedure that the host software uses to modify the selected bits is as follows:

1. Sets bits 2–1 of PCI configuration register 92h (function 0) to 01b for interrupt mode selection.
2. Writes to the upper four bits of ExCA register 05h/45h/805h for desired CSC routing for each socket. (Restrictions are placed on interrupt routing with ISA IRQ signaling; only ten IRQs are valid in this mode.)
3. If a PC Card is installed in the socket and requires functional interrupts, writes to the lower nibble of ExCA register 03h/43h/803h for the desired functional interrupt routing for the socket. (Note the restrictions placed on interrupt routing with ISA IRQ signaling.)
4. Using Table 2, writes to the appropriate mask register bits to enable interrupt generation for desired events.

For card-removal events, the host software masks any functional interrupts that were set for that socket.

For card-insertion events, the host software reconfigures the mask and routing registers to support the new card requirements.

2.3 PCI Interrupts

> **NOTE:** PCI interrupts can be used with ISA interrupts. Unused $\overline{\text{INTA}}$ and $\overline{\text{INTB}}$ lines should be pulled high with 43-kΩ pullup resistors.

The PCI1131 also supports interrupt signaling that is compliant with the *PCI Local Bus Specification* [3]. Consistent with this specification, the PCI1131 can use one PCI interrupt for each of its functions. As a result, $\overline{\text{INTA}}$ is used for PC Card socket A interrupts, and $\overline{\text{INTB}}$ for socket B. These terminals are found on the PCI1131 at pins 154 and 155, respectively, and are dual-function pins with the ISA-mode interrupts IRQ3 and IRQ4. When the PCI1131 is configured for PCI interrupt signaling, these pins behave as open-drain PCI interrupts. Systems that prefer a single interrupt line from the PCI1131 can connect these two interrupt terminals.

PCI configuration register offset 91h must be written to, to route CSC and functional interrupts from each socket. The step-by-step events that the host software must follow to successfully configure the PCI1131 for PCI signaling are listed in the following procedure. These steps assume that the system powered up and that the PCI reset ($\overline{\text{RSTIN}}$) pin is set high (deasserted). If only selected bits of a register are to be modified, the host software leaves the remaining register bits unchanged first by reading the current contents of the register, then by modifying the desired bits, and last by writing the new value back to the register.

The procedure that the host software uses to modify the selected bits is as follows:

1. Sets bit 5 of PCI configuration register 91h (function 0) to a value of 1 (enabled).
2. Sets bit 3 of PCI configuration register 91h (functions 0 and 1 separately) to route CSC interrupts to $\overline{\text{INTA}}$ (for socket A) or $\overline{\text{INTB}}$ (for socket B).
3. If a PC Card is installed in the socket and requires functional interrupts, writes to bit 4 of the PCI card control register 91h (for the socket in question) to route functional interrupts from the PC Card to $\overline{\text{INTA}}$ (for socket A) or $\overline{\text{INTB}}$ (for socket B).
4. Using Table 2, writes to the appropriate mask register bits to enable interrupt generation for the desired events.

For card-removal events, the host software masks any functional interrupts that were set for that socket.

For card-insertion events, the host software reconfigures the mask and routing registers to support the new card requirements.

2.4 Serial IRQ Signaling

The serial interrupt protocol implemented in the PCI1131 uses a single PCI1131 terminal to communicate all interrupt-status information to the host interrupt controller. The protocol defines a serial packet consisting of a start cycle, a stop cycle, and multiple interrupt cycles. All data in the packet are synchronous with the PCI clock (PCLK). The duration of the stop and interrupt cycles is a fixed number of clock periods, but the start cycle is variable (four, six, or eight clock periods). This allows the serial packet to retain coherence on either side of a PCI–PCI bridge.

Figure 4 and Figure 5 illustrate how the serial IRQ protocol works. Figure 4 shows the start cycle and the initial IRQ sampling periods. Figure 5 shows the final IRQ sampling periods and the stop cycle. The intermediate IRQ sampling periods are not shown, but the sampling periods occur in ascending IRQ order: IRQ0, IRQ1, SMI, IRQ3, IRQ4 ... IRQ15, and $\overline{\text{IOCHK}}$. The IRQ signals are active high. In the following illustrations, IRQ1 and IRQ15 are sampled deasserted. The stop cycle must occur after the $\overline{\text{IOCHK}}$ period, but may be extended to allow more sampling periods for platform-specific functions.

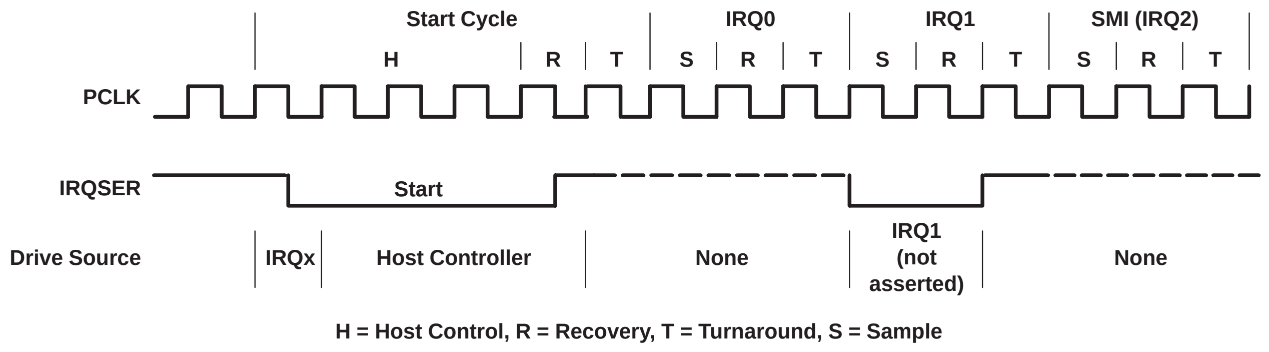

Figure 4. Serial Interrupt Timing – Start Cycle and IRQ Sampling Periods

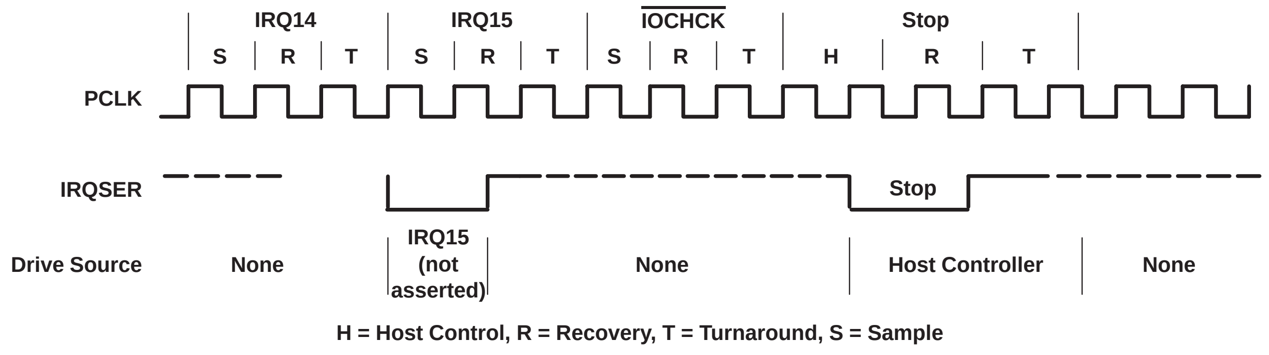

Figure 5. Serial Interrupt Timing – Stop Cycle

In a system using the serialized IRQ protocol, the host software must configure the PCI1131 to use serialized IRQs by setting bits 2–1 of the device control register at offset 92h to 10b**.** The step-by-step events that host software must follow to successfully configure the PCI1131 for serialized IRQ signaling are listed in the following procedure. These steps assume that the system powered up and that the PCI reset ($\overline{\text{RSTIN}}$) pin is set high (deasserted). If only selected bits of a register are modified, the host software must leave the remaining register bits unchanged first by reading the current contents of the register, then by modifying the desired bits, and last by writing the new value back to the register.

The procedure that the host software uses to modify the selected bits is as follows:

1. Set bits 2–1 of PCI device control register 92h (function 0) to 10b.
2. Writes to the upper nibble of ExCA register 05h/45h/805h for desired CSC routing for each socket. (Note that all 15 IRQs are available for routing when serialized IRQ signaling is selected.)
3. If a PC Card is installed in the socket and requires functional interrupts, writes to the lower nibble of ExCA register 03h/43h/803h for the desired functional interrupt routing for the socket.
4. Using Table 2, writes to the appropriate mask register bits to enable interrupt generation for desired events.

For card-removal events, the host software masks any functional interrupts that were set for that socket.

For card-insertion events, the host software reconfigures the mask and routing registers to support the new card requirements.

3 References

1. *PCMCIA CardBus Specification*, PCMCIA, San Jose, Calif.
2. *Serialized IRQ Support for PCI Systems*, *Revision 6.0*, September 1, 1995, PCI Special Interest Group (SIG), Portland, Ore.
3. *PCI Local Bus Specification, Revision 2.1,* PCI SIG, Portland, Ore.