Connecting the PCI Interface of the RC32438 to a CardBus Interface

Application Note
AN-442

Notes

By Fred Santilo

Revision History

August 26, 2003: Initial publication.

Background

The RC32438 is a member of the IDT™ Interprise™ family of PCI integrated communications processors. It is a high performance, general-purpose integrated processor that boasts industry-leading aggregate bandwidth. Featuring a MIPS CPU core, this device also includes a memory controller supporting double data rate (DDR) memory and a PCI (Peripheral Component Interconnect) interface capable of sustaining up to 160 MBps data transfer across the bus and featuring an on-chip PCI arbiter to simplify the design of embedded systems. The RC32438 also includes two 10/100Mbps Ethernet ports, providing MII interfaces off-chip. Appropriate applications include layer 3 Ethernet Switches, office gateways, wireless access points, virtual private network (VPN) systems, and storage area networks.

This application note describes how to connect and configure a CardBus Bus Master device and a standard PCI Bus Master device to the PCI interface of the RC32438. Note that the CardBus device will not be able to utilize its Hot Insertion/Removal or Clock Run capabilities, since they are not defined in the PCI 2.2 specification.

PCI Interface

RC32438 PCI Pin Description

The PCI interface provides a high performance peripheral bus solution. The RC32438 PCI interface is PCI Revision 2.2 compliant and can operate at clock speeds from 16MHz to 66MHz. Table 1 lists the PCI signals for the RC32438. Recommended or required external pull-up and pull-down resistors are 10 kΩ.

For a detailed description of the RC32438 PCI signals and their functionality, refer to the 79RC32438 User Reference Manual located on the company's web site at www.idt.com.

Name	Type	Description
PCIAD[31:0]	I/O	**PCI Multiplexed Address/Data Bus** Address is driven by Bus Master during initial PCIFRAMEN assertion. Data is then driven by the Bus Master during writes, or the Bus Target during reads.
PCICBEN[3:0]	I/O	**PCI Multiplexed Command/Byte Enable Bus** PCI command is driven by the Bus Master during the initial PCIFRAMEN assertion. Byte Enables are driven by the Bus Master during subsequent data phase(s).
PCIPAR	I/O	**PCI Parity** Even parity of the PCIAD[31:0] bus. Driven by the Bus Master during Address and Write Data phases. Driven by the Bus Target during the Read Data phase.
PCIFRAMEN	I/O	**PCI Frame** Driven by the Bus Master. Assertion indicates the beginning of a bus transaction. De-assertion indicates the last data.
PCITRDYN	I/O	**PCI Target Ready** Driven by the Bus Target to indicate the current data can complete.

Table 1 RC32438 PCI Pin Description (Page 1 of 3)

Notes

Name	Type	Description
PCIIRDYN	I/O	**PCI Initiator Ready** Driven by the Bus Master to indicate that the current data can complete.
PCISTOPN	I/O	**PCI Stop** Driven by the Bus Target to terminate the current bus transaction.
PCIIDSELP	Input	**PCI Initialization Device Select** Uses PCIREQN[1] pin. See Chapter 10, PCI Bus Interface, in the RC32438 User Reference Manual.
PCIPERRN	I/O	**PCI Parity Error** This signal is asserted by the receiving Bus Agent 2 clocks after the data is received, if a parity error is detected.
PCISERRN	I/O Open-collector	**System Error** External pull-up resistor is required. This signal is asserted by any agent to indicate an address parity error, data parity during a Special Cycle command, or any other system error.
PCICLK	Input	**PCI Clock** Clock used for all PCI Bus transactions.
PCIRSTN	Input	**PCI Reset** In Host mode this signal is asserted by the RC32438 to generate a PCI reset. In Satellite mode the assertion of this signal initiates a warm reset.
PCIDEVSELN	I/O	**PCI Device Select** This signal is driven by a bus target to indicate that the target has decoded the present address as one of its own address spaces.
PCIREQN[3:0]	Input	**PCI Bus Request.** **In PCI host mode with internal arbiter:** These signals are inputs whose assertion indicates to the internal RC32438 arbiter that an agent desires ownership of the PCI bus. **In PCI host mode with external arbiter:** PCIREQN[0]: asserted by the RC32438 to request ownership of the PCI bus. PCIREQN[3:1]: unused and driven high. **In PCI satellite mode:** PCIREQN[0]: this signal is asserted by the RC32438 to request use of the PCI bus. PCIREQN[1]: PCIIDSELP and is used as a chip select during configuration read and write transactions. PCIREQN[3:2]: unused and driven high. **Note**: When the GPIO register is programmed in the alternate function mode for bits GPIO [24] and [27], these bits become PCIREQN [4] and [5] respectively.
GPIO[24]	I/O	**General Purpose I/O.** This pin can be configured as a general purpose I/O pin. Alternate function pin name: PCIREQN[4] Alternate function: PCI Request 4
GPIO[27]	I/O	**General Purpose I/O.** This pin can be configured as a general purpose I/O pin. Alternate function pin name: PCIREQN[5] Alternate function: PCI Request 5

Table 1 RC32438 PCI Pin Description (Page 2 of 3)

Notes

Name	Type	Description
PCIGNTN[3:0]	Output	**PCI Bus Grant**. **In PCI host mode with internal arbiter:** The assertion of these signals indicates to the agent that the internal RC32438 arbiter has granted the agent access to the PCI bus. **In PCI host mode with external arbiter:** PCIGNTN[0]: asserted by an external arbiter to indicate to the RC32438 that access to the PCI bus has been granted. PCIGNTN[3:1]: unused and driven high. **In PCI satellite mode:** PCIGNTN[0]: this signal is asserted by an external arbiter to indicate to the RC32438 that access to the PCI bus has been granted. PCIGNTN[1]: this signal takes on the alternate function of PCIEECS and is used as a PCI Serial EEPROM chip select. PCIGNTN[3:2]: unused and driven high. **Note**: When the GPIO register is programmed in the alternate function mode for bits GPIO [26] and [28], these bits become PCIGNTN [4] and [5] respectively.
GPIO[26]	I/O	**General Purpose I/O.** This pin can be configured as a general purpose I/O pin. Alternate function pin name: PCIGNTN[4] Alternate function: PCI Grant 4
GPIO[28]	I/O	**General Purpose I/O.** This pin can be configured as a general purpose I/O pin. Alternate function pin name: PCIGNTN[5] Alternate function: PCI Grant 5
PCILOCKN	Input	**PCI Lock** This signal is asserted by an external Bus Master to indicate that an exclusive operation is occurring.
GPIO[30]	I/O	**General Purpose I/O.** This pin can be configured as a general purpose I/O pin. Alternate function pin name: PCIMUINTN Alternate function: PCI Messaging unit interrupt output

Table 1 RC32438 PCI Pin Description (Page 3 of 3)

RC32438 PCI Configuration Space Header Registers

The RC32438 utilizes the Type 00h Configuration Space Header that is defined in the PCI Revision 2.2 specification. The Configuration Space Header contains 256-bytes (16 32-bit words) which are divided into two layout regions.

The first region contains 16-bytes and is mandatory. The 16-bytes are laid out in a predefined manner for all Type 00h PCI devices.

The second region contains the remaining 240-bytes of the Configuration Space Header, with its layout being dependent on the base function of the device. A PCI device must implement the appropriate registers for the functions it supports in the defined location and with the defined functionality. Figure 1 shows the format of a 256-byte PCI Type 00h PCI Configuration Space Header.

Notes

31	16	15	0	
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Latency Timer	Cache Line Size	0Ch
Base Address 0				10h
Base Address 1				14h
Base Address 2				18h
Base Address 3				1Ch
Base Address 4				20h
Base Address 5				24h
CardBus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved				34h
Reserved				38h
Max Latency	Min Grant	Interrupt Pin	Interrupt Line	3Ch

Figure 1 Type 00h PCI Configuration Space Header

PCI Configuration Cycle

A PCI device's PCI Configuration Space Header is configured via Configuration Read and Write cycles, which are software driven. Each device sitting on the PCI interface is connected to a unique Initialization Device Select (PCIIDSELN) signal that acts as a Chip Select signal. A PCI device is the Target of a read or write command when its PCIIDSELP is asserted and its Address Data [1:0] bus contains a 00b value during the address phase of the transaction. The Target device then decodes PCIAD[7:0] and Command/Byte Enable[3:0] (PCICBEN[3:0]) to determine which register within its Configuration Header Space is to be accessed. Configuration cycles which do not receive a response from any device on the PCI interface are treated as Master Aborts.

A simplified method of evaluating a PCI Configuration cycle is to split it into two parts, an address phase and a data phase. The address phase is used to select a particular device and the data phase is used for the actual transfer of data bytes between the Master and the Target.

The address phase consists of the Configuration Host asserting PCIFRAMEN low, placing an address on the PCIAD[31:0] bus, asserting the appropriate PCIIDSELP high, and issuing either a Configuration Read or a Write command on the PCICBEN[3:0] bus. The Target device will detect its PCIIDSELP, which is unique for each device on the PCI interface, being asserted. The address phase of the transaction is now complete.

The data phase continues where the address phase ends. The Configuration Host will assert Initiator Ready (PCIIRDYN) low and assert the desired byte enables on the PCICBEN[3:0] bus to indicate it is ready to start the data transfer. The Target will assert Target Ready (PCITRDYN) low and assert its PCIDEVSELN signal low to indicate it is ready for the data transfer. The PCIAD[31:0] is driven by the Target device during a Configuration Read and is driven by the Master (Configuration Host) for a Configuration Write. All signals are de-asserted to their non active state once the data transfer is complete.

Notes

Note that this is a simplified description and does not include information such as wait states and bus turnaround cycles. Refer to the PCI Revision 2.2 specification for detailed information on PCI Configuration Cycle timing diagrams and protocol.

Typical PCI Read or Write Transactions

Similar to a PCI Configuration cycle, a PCI Read or Write transaction is split into two parts, the address phase and the data phase. The address phase is used to communicate which device is the Target for the transaction, and the data phase takes place during the actual transfer of data bytes.

The address phase for a Read or Write command is exactly the same. A Master (Initiator) device starts the address phase of the transaction by asserting its Request (PCIREQN) signal to request ownership of the PCI bus. Each PCI device residing on the PCI interface is connected to a unique PCIREQN signal. The PCI Arbiter will detect the assertion of the PCIREQN signal and will assert the appropriate Grant (PCIGNTN) signal, which is also unique for each device residing on the interface. The PCIGNTN signal is an acknowledgement to the Master that it can safely take possession of the PCI bus. Once the Master takes ownership of the bus, it will assert its PCIFRAMEN signal along with the appropriate address on the PCIAD[31:0] bus. The Master will also place the desired Command Type on the PCICBEN[3:0] bus at this time to indicate the type of command that will be executed. When PCIFRAMEN is asserted, each Target device samples the PCIAD[31:0] bus to determine if it is the Target the Master is trying to communicate with. The appropriate Target device will then assert the Device Select (PCIDEVSELN) signal to acknowledge that it is the owner of the address. The address phase is now complete.

The data phase continues where the address phase leaves off. The Master will assert the PCIIRDYN signal to indicate to the Target that it is ready to start the data transfer. The Target responds by asserting the PCITRDYN signal indicating it is also ready to receive the data. Data is transferred when PCIFRAMEN, PCIIRDYN, PCITRDYN, and PCIDEVSELN are all asserted at the same time. The difference between a Read or Write data phase is whether the Master or Target drives the PCIAD[31:0] bus during the data transfer. During a Read transaction, the Target drives the PCIAD[31:0] bus; during a Write transaction, the Master drives the PCIAD[31:0] bus.

Once the data transfer is complete, the PCI bus is released to the arbiter or parked on a predefined PCI Bus Master device.

Note that this is a simplified explanation of a PCI Read or Write transaction and does not include information such as wait states and bus turnaround cycles. Refer to the PCI Revision 2.2 specification for more detailed information on commands, protocol, and timing specifications.

PCI Interrupts

In host mode, the RC32438 does not provide a dedicated interrupt input line, such as INTA. However, the PCI Messaging Unit Interrupt signal (PCIMUINTN) available via the alternate function of GPIO[30] can be used as a dedicated PCI interrupt line. It operates in both satellite and host modes and tri-states when not asserted (i.e., it acts as an open collector output). Please refer to the General Purpose I/O Controller and the PCI Bus Interface chapters of the RC32438 Users Reference Manual for more detailed information.

CardBus Interface

CardBus Pin Description

CardBus integrates the high performance of a 32-bit PCI device in a low-power, small form factor with hot insertion capabilities that is backward compatible with a PCMCIA 16-bit PC card. In a typical application, the CardBus device has a point to point connection with the CardBus bridge it is attached to.

The CardBus interface shares the same bus commands and bus command protocols as a PCI Revision 2.2 device. However, it does differ in the way it connects to the system, how it is configured by the system host, and in its interrupt generation. Signals associated with a CardBus device are listed in Table 2. Recommended or required external pull-up and pull-down resistors are 10 kΩ.

Notes

This application note assumes the CardBus device will be used as a Master and a Target device, thus the Pin Description includes the Bus Request and Bus Grant signals.

Name	Type	Description
CAD[31:0]	I/O	**Multiplexed Address/Data Bus** Address driven by Bus Master during initial CFRAME# assertion, then the Data is driven by the Bus Master during writes; or the Data is driven by the Bus Target during reads.
CC/BE[3:0]#	I/O	**Multiplexed Command/Byte Enable Bus Negated** Command (not negated) Bus driven by the Bus Master during the initial FRAME# assertion. Byte Enable Negated Bus driven by the Bus Master during the data phase(s).
CPAR	I/O	**Parity** Even parity of the CAD[31:0] bus. Driven by Bus Master during Address and Write Data phases. Driven by the Bus Target during the Read Data phase.
CFRAME#	I/O	**Frame Negated** Driven by the Bus Master. Assertion indicates the beginning of a bus transaction. De-assertion indicates the last data.
CTRDY#	I/O	**Target Ready Negated** Driven by the Bus Target to indicate the current data can complete.
CIRDY#	I/O	**Initiator Ready Negated** Driven by the Bus Master to indicate that the current data can complete.
CSTOP#	I/O	**Stop Negated** Driven by the Bus Target to terminate the current bus transaction.
CPERR#	I/O	**Parity Error Negated** Driven by the receiving Bus Agent 2 clocks after the data is received, if a parity error occurs.
CSERR#	I/O	**System Error Negated** Driven by any agent to indicate an address parity error, data parity during a Special Cycle command, or any other system error.
CCLK	Input	**Clock** Clock for Bus transactions. Uses the rising edge for all timing references.
CRST#	Input	**Reset Negated** Resets all related logic.
CDEVSEL#	I/O	**Device Select Negated** Driven by the target to indicate that the target has decoded the present address as a target address.
CREQ#	Output	**Bus Request Negated** An output indicating a request from this device.
CGNT#	Input	**Bus Grant Negated** An input indicating a grant to this device.
CINT#	Output	**Interrupt Negated** Interrupt request signal.

Table 2 CardBus Pin Description (Page 1 of 2)

Notes

Name	Type	Description
CBLOCK#	Input	**Lock Negated** Driven by the Bus Master to indicate that an exclusive operation is occurring.
CCD[2:1]#	Output	**Card Detection Negated** Indicates insertion and removal of the CardBus device. Require external pull-up resistor.
CVS[2:1]	I/O	**Voltage Sense** Indicates VCC requirements.
CAUDIO#	Output	**Card Audio Negated** Digital audio output.
CSTSCHG#	Output	**Card Status Change Negated** Indicates system change or used as a wake up.

Table 2 CardBus Pin Description (Page 2 of 2)

CardBus Card Configuration Header Registers

The CardBus Configuration Header Register space is similar to the Type 00H Configuration Space Header defined in the PCI Revision 2.2 specification. To be compatible with the RC32438 device, the CardBus device must have the CardBus Type 00H layout. The Type 00H CardBus Configuration Header space contains 256-bytes (16 32-bit words), which are divided into two layout regions.

The first region contains 16-bytes and is mandatory. The 16-bytes are laid out in a predefined manner similar to the Type 00H PCI devices. The difference between the two header spaces is that the Vendor Identification and Class Code fields of the PCI header space are not required fields for the CardBus device; they are defined as Allocated. Allocated registers maintain compatibility with environments other than CardBus and may be implemented as readable fields.

The second region contains the remaining 240-bytes of the Type 00H CardBus Configuration Header space. This region is device-specific and must always exist. It is similar to the PCI register space in that a CardBus device must implement the appropriate registers for the functions it supports in the defined location with the defined functionality. Unused registers must contain a zero value. Differences between the second region of the PCI header space and the second region of the CardBus header space are that the CardBus header space does not contain the Maximum Latency, Minimum Grant, and Interrupt Line fields of the PCI header space located at offset 0x3C. Instead, these fields are defined as Allocated fields. Also, the Capability Pointer field of the CardBus header space at offset 0x34 is a reserved field in the PCI layout.

Figure 2 shows the format of a 256-byte PCI Type 00h PCI Configuration Space Header.

Notes

31	16	15	0	
Allocated		Allocated		00h
Status		Command		04h
Allocated			Allocated	08h
BIST	Header Type	Latency Timer	Cache Line Size	0Ch
Base Address 0				10h
Base Address 1				14h
Base Address 2				18h
Base Address 3				1Ch
Base Address 4				20h
Base Address 5				24h
CardBus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved				34h
Reserved				38h
Allocated	Allocated	Interrupt Pin	Allocated	3Ch

Figure 2 CardBus Configuration Header Space

CardBus Configuration Cycle

CardBus Configuration Cycles are software driven and are very similar to the PCI configuration cycles with the following exception: a CardBus device does not posses a PCIIDSELP signal since it is typically connected to the PCI interface via a PCI to CardBus Bridge. Because it is a point to point connection to a CardBus Bridge, a CardBus device responds to all configuration cycles passed by the bridge. Thus, the CardBus Bridge is configured to filter and only pass configuration cycles intended for the CardBus device that is connected to it.

The CardBus Configuration Header Space is configured via Configuration Read and Write cycles, which mimic the PCI configuration cycles. During the address phase of the CardBus configuration cycle, CAD[10:8] selects the device function while CAD[7:2] is used to specify the appropriate double word of the 256-byte region.

Refer to the PC Card Standard specification for detailed information pertaining to CardBus configuration cycle commands, timing, and protocol.

Typical CardBus Read or Write Transactions

CardBus devices support the same commands, timing, and protocols as defined in the PCI Revision 2.2 specification. Refer to the Typical PCI Read or Write Transactions section of this document or the PC Card specification for more detailed information.

Connecting the PCI Interface of the RC32438 to a CardBus Device

As mentioned previously, the significant differences between PCI and CardBus protocols are the manner in which a device is detected during the configuration process, the number of interrupt lines, Hot Insertion/Removal capabilities, and the Clock Run option.

Notes

In a PCI application, the device is recognized by its PCIIDSELP signal, and in the CardBus application, the CardBus device sits behind a CardBus Bridge that filters its configuration cycles. External logic, in the form of a buffer with an output enable, must be incorporated to control the Configuration Cycles seen by the CardBus device in order to properly configure the CardBus device. The buffer can be in the form of a quick-switch or a single gate buffer device that is enabled via one of the RC32438 GPIO pins, which acts as a PCIIDSELP and is used to control the assertion of PCIFRAMEN to the CardBus device during the configuration and non-configuration cycles. This implies that the configuration software must know which of the PCI slots the CardBus device resides in and which configuration cycles are to be passed to the CardBus device.

The assertion and de-assertion of the GPIO signal is as follows:

1. The GPIO signal is asserted low to enable the buffer for all configuration cycles meant for the CardBus device.
2. The GPIO signal is asserted high to disable the buffer for all configuration cycles not meant for the CardBus device.
3. The GPIO signal is asserted low to enable the buffer for all non configuration cycles.

The single interrupt line, CINT#, of the CardBus device is connected to the RC32438 GPIO[30] signal (PCIMUINTN). The interrupt signals from multiple PCI devices, such as in Figure 3, can be connected to the RC32438 PCIMUINTN signal, since it will tri-state when not being driven.

The Hot Insertion/Removal capabilities of the CardBus device are not supported by the RC32438. Therefore, this function cannot be used and the configuration software must know the card type of the CardBus device prior to configuring it. The CCD1#, CCD2#, CVS1, and CVS2 signals of the CardBus device are left open, unconnected, since they are all output signals from the CardBus device.

The Clock Run option of the CardBus specification is not supported by the RC32438. Therefore, the CCLKRUN# signal is tied to ground to disable this option.

Figure 3 shows the RC32438 device connected to a PCI Bus Master device and a CardBus Bus Master device. The CardBus device is shown as a Bus Master device since it is not likely that a CardBus device would be a Target only device. However, a Target only CardBus device would be connected to the RC32438 in the same manner, excluding the Bus Request and Bus Grant signals that the Target CardBus device would not have.

Notes

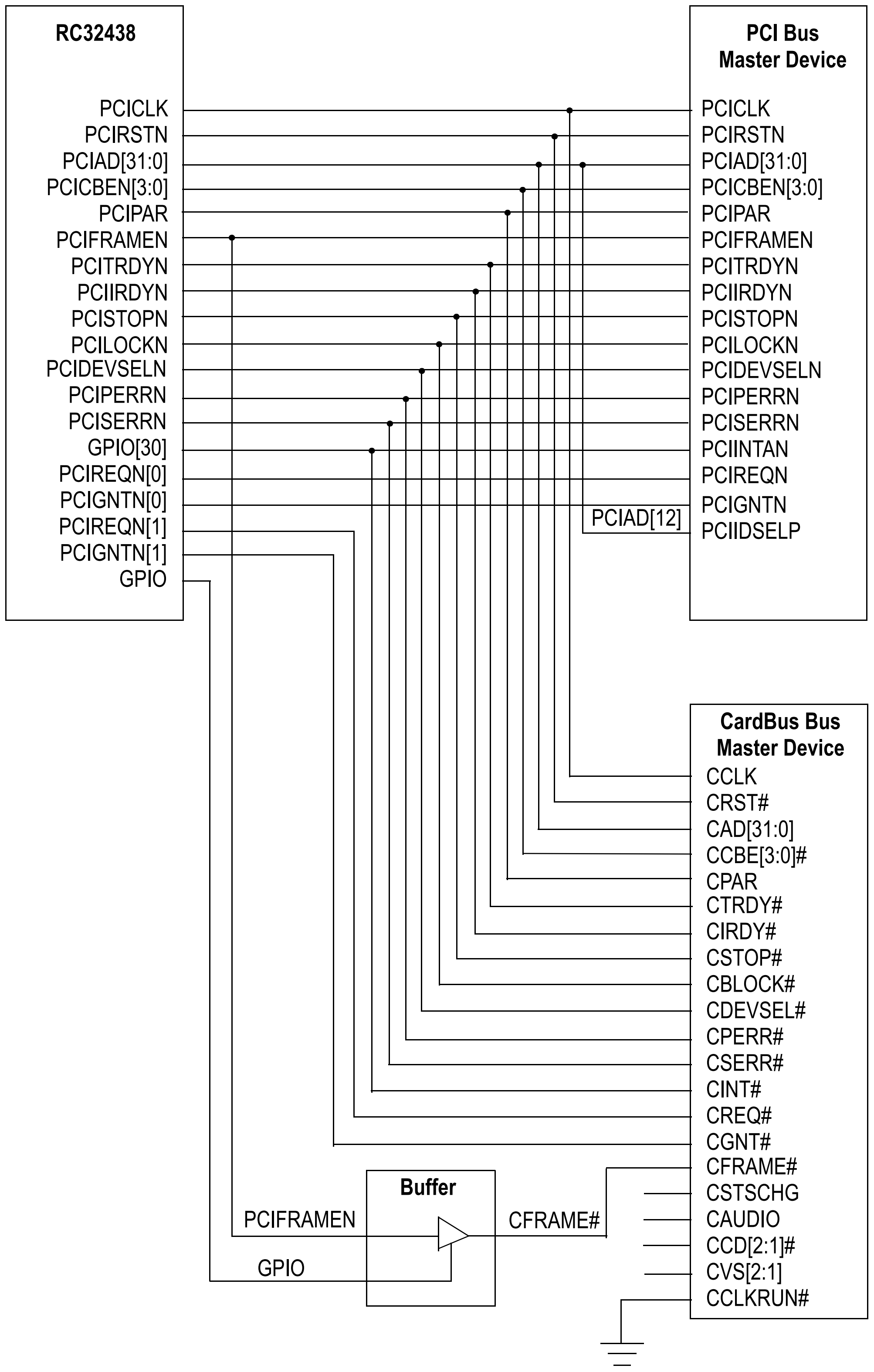

Figure 3 Connection Between the RC32438, a PCI Bus Master Device, and a CardBus Bus Master Device

Notes

Conclusion

The PCI and CardBus protocols are very similar. The difference is the manner in which they are detected and configured by the host device. Using a buffer with an output enable to filter the CardBus configuration cycles allows a designer to connect a CardBus device to the RC32438 processors without implementing a PCI to CardBus Bridge device in the system, thus reducing design time, board cost, and board real estate.

References

79RC32438 User Reference Manual.

PC Card Standard specification.

PCI Revision 2.2 specification.

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks