Connecting the PCI Interface on the RC3233x Series to a CardBus Interface

Application Note AN-396

Notes

By Fred Santilo

Revision History

November 18, 2002: Initial publication.

Background

The RC3233x series of integrated processors combine a 32-bit MIPS instruction set architecture (ISA) CPU core with a number of on-chip peripherals to enable direct connection to boot memory, main memory, I/O devices, and PCI devices. The RC3233x devices also include system logic for DMA, reset, interrupts, timers, and UARTs. The integrated components reduce board real estate, design time, and system cost.

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 RC3233x. 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 2.1 specification.

PCI Interface

RC3233x PCI Pin Description

The PCI (Peripheral Component Interconnect) interface provides a high performance peripheral bus solution. The RC3233x PCI interface is PCI Revision 2.1 compliant and can operate at clock speeds up to 66MHz. Table 1 lists the PCI signals for the RC3233x. The designation RC3233x in the Device column means the signal applies to all three devices: RC32332, RC32333, and RC32334. Recommended or required external pull-up and pull-down resistors are 10 kΩ.

For a detailed description of the RC3233x PCI signals and their functionality, refer to the IDT79RC3233x User Reference Manual located on the company's web site at www.idt.com.

Name	Device	Type	Reset State Status	Description
pci_ad[31:0]	RC3233x	I/O	Z	**PCI Multiplexed Address/Data Bus** Address driven by Bus Master during initial frame_n assertion, and then the Data is driven by the Bus Master during writes; or the Data is driven by the Bus Slave during reads.
pci_cbe_n[3:0]	RC3233x	I/O	Z	**PCI Multiplexed Command/Byte Enable Bus** Command (not negated) Bus driven by the Bus Master during the initial frame_n assertion. Byte Enable Negated Bus driven by the Bus Master during the data phase(s).
pci_par	RC3233x	I/O	Z	**PCI Parity** Even parity of the pci_AD[31:0] bus. Driven by Bus Master during Address and Write Data phases. Driven by the Bus Slave during the Read Data phase.

Table 1 RC32334 PCI Pin Description (Page 1 of 3)

Notes

Name	Device	Type	Reset State Status	Description
pci_frame_n	RC3233x	I/O	Z	**PCI Frame Negated** Driven by the Bus Master. Assertion indicates the beginning of a bus transaction. De-assertion indicates the last datum.
pci_trdy_n	RC3233x	I/O	Z	**PCI Target Ready Negated** Driven by the Bus Slave to indicate the current datum can complete.
pci_irdy_n	RC3233x	I/O	Z	**PCI Initiator Ready Negated** Driven by the Bus Master to indicate that the current datum can complete.
pci_stop_n	RC3233x	I/O	Z	**PCI Stop Negated** Driven by the Bus Slave to terminate the current bus transaction.
pci_idsel	RC3233x	Input		**PCI Initialization Device Select** Uses pci_req_n[2] pin. See Chapter 12, PCI Interface Controller, in the RC3233x User Reference Manual.
pci_perr_n	RC3233x	I/O	Z	**PCI Parity Error Negated** Driven by the receiving Bus Agent 2 clocks after the data is received, if a parity error occurs.
pci_serr_n	RC3233x	I/O Open-collector	Z	**System Error** External pull-up resistor is required. Driven by any agent to indicate an address parity error, data parity during a Special Cycle command, or any other system error.
pci_clk	RC3233x	Input		**PCI Clock** Clock for PCI Bus transactions. Uses the rising edge for all timing references.
pci_rst_n	RC3233x	Input	L	**PCI Reset Negated** Host mode: Resets all PCI related logic. Satellite mode: with boot from PCI mode: Resets all PCI related logic and also warm resets the 3233x.
pci_devsel_n	RC3233x	I/O	Z	**PCI Device Select Negated** Driven by the target to indicate that the target has decoded the present address as a target address.
pci_req[2]#	RC3233x	Input	Z	**PCI Bus Request #2 Negated** Requires external pull-up. Host mode: pci_req_n[2] is an input indicating a request from an external device. Satellite mode: used as pci_idsel pin which selects this device during a configuration read or write. Alternate function: pci_idsel (satellite).
pci_req[1]#	RC32333 and RC32334	Input	Z	**PCI Bus Request #1 Negated** Requires external pull-up. Host mode: pci_req_n[1] is an input indicating a request from an external device. Alternate function: Unused (satellite).

Table 1 RC32334 PCI Pin Description (Page 2 of 3)

Notes

Name	Device	Type	Reset State Status	Description
pci_req[0]#	RC3233x	I/O	Z	**PCI Bus Request #0 Negated** Requires external pull-up for burst mode. Host mode: pci_req_n[0] is an input indicating a request from an external device. Satellite mode: pci_req_n[0] is an output indicating a request from this device.
pci_gnt_n[2]	RC3233x	Output	Z[1]	**PCI Bus Grant #2 Negated** Recommend external pull-up. Host mode: pci_gnt_n[2] is an output indicating a grant to an external device. Satellite mode: pci_gnt_n[2] is used as the pci_inta_n output pin. Alternate function: pci_inta_n (satellite).
pci_gnt_n[1] / pci_eeprom_cs	RC32333 and RC32334	I/O	X for 1 pci clock then H[2]	**PCI Bus Grant #1 Negated** Recommend external pull-up. Host mode: pci_gnt_n[2:1] are outputs indicating grants to external devices. Satellite mode: Used as pci_eeprom_cs output pin for Serial Chip Select for loading PCI Configuration Registers in the RC32334 Reset Initialization Vector PCI boot mode. Defaults to the output direction at reset time. 1st Alternate function: pci_eeprom_cs (satellite). 2nd Alternate function: PIO[11].
pci_gnt_n[1] (can only be used as alternate function pci_eeprom_cs)	RC32332	I/O	X for 1 pci clock then H[3]	**PCI Bus Grant #1 Negated** Recommend external pull-up. Host mode: not used as pci_gnt_n[1]. Must be used as alternate function PIO[7]. Satellite mode: Not used as pci_gnt_n[1]. Used as pci_eprom_cs output pin for Serial Chip Select for loading PCI Configuration Registers in the RC32332 Reset Initialization Vector PCI boot mode. Defaults to the output direction at reset time. 1st Alternate function: pci_eeprom_cs (satellite). 2nd Alternate function: PIO[7].
pci_gn_nt[0]	RC3233x	I/O	Z	**PCI Bus Grant #0 Negated** Host mode: pci_gnt_n[0] is an output indicating a grant to an external device. Recommend external pull-up. Satellite mode: pci_gnt_n[0] is an input indicating a grant to this device. Require external pull-up.
pci_inta_n	RC3233x	Output Open-collector	Z	**PCI Interrupt #A Negated** Uses pci_gnt_n[2]. See Chapter 12, PCI Interface Controller, in the RC3233x User Reference Manual.
pci_lock_n	RC3233x	Input		**PCI Lock Negated** Driven by the Bus Master to indicate that an exclusive operation is occurring.

Table 1 RC32334 PCI Pin Description (Page 3 of 3)

1. Z in host mode; L in satellite non-boot mode; Z in satellite boot mode.

2. H in host mode; L in satellite non-boot and boot modes. X = unknown.

3. H in host mode; L in satellite non-boot and boot modes. X = unknown.

Notes

RC3233x PCI Configuration Space Header Registers

The RC3233x utilizes the Type 00h Configuration Space Header that is defined in the PCI Revision 2.1 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.

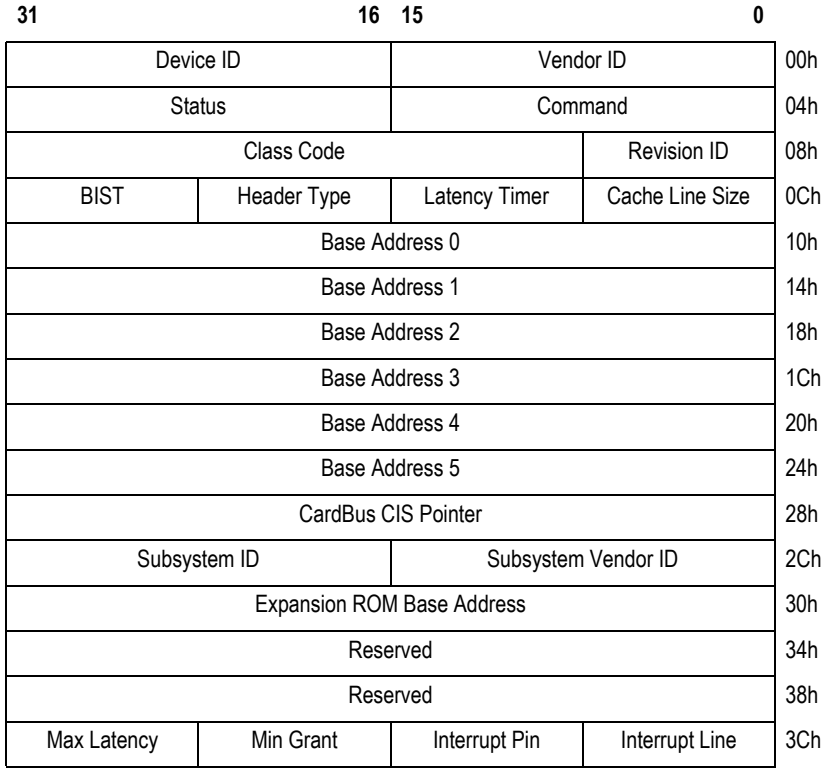

Figure 1 Type 00h PCI Configuration Space Header

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 (idsel) signal that acts as a Chip Select signal. A PCI device is the Target of a read or write command when its idsel 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 ad[7:0] and Command/Byte Enable [3:0] (cbe_n[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.

Notes

The address phase consists of the Configuration Host asserting frame_n low, placing an address on the ad[31:0] bus, asserting the appropriate idsel high, and either issuing a Configuration Read or Write command on the cbe_n[3:0] bus. The Target device will detect its idsel, 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 were the address phase ends. The Configuration Host will assert Initiator Ready (irdy_n) low and assert the desired byte enables on the cbe_n[3:0] bus to indicate it is ready to start the data transfer. The Target will assert Target Ready (trdy_n) low and assert its devsel_n signal low indicating it is ready for the data transfer. The ad[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.

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 2.1 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 (req_n) signal to request ownership of the PCI bus. Each PCI device residing on the PCI interface is connected to a unique req_n signal. The PCI Arbiter will detect the assertion of the req_n signal and will assert the appropriate Grant (gnt_n) signal, which is also unique for each device residing on the interface. The gnt_n 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 frame_n signal along with the appropriate address on the ad[31:0] bus. The Master will also place the desired Command Type on the cbe_n[3:0] bus at this time to indicate the type of command that will be executed. Each Target device samples the ad[31:0] bus when frame_n is asserted to determine if it is the Target the Master is trying to communicate with. The appropriate Target device will then assert the Device Select (devsel_n) 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 irdy_n signal to indicate to the Target that it is ready to start the data transfer. The Target responds by asserting the trdy_n signal indicating it is also ready to receive the data. Data is transferred when frame_n, irdy_n, trdy_n, and devsel_n are all asserted at the same time. The difference between a Read or Write data phase is whether the Master or Target drives the ad[31:0] bus during the data transfer. During a Read transaction, the Target drives the ad[31:0] bus; during a Write transaction, the Master drives the ad[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 included information such as wait states and bus turnaround cycles. Refer to the PCI Revision 2.1 specification for more detailed information on commands, protocol, and timing specifications.

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 2.1 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, and then the Data is driven by the Bus Master during writes; or the Data is driven by the Bus Slave 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 Slave 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 datum.
CTRDY#	I/O	**Target Ready Negated** Driven by the Bus Slave to indicate the current datum can complete.
CIRDY#	I/O	**Initiator Ready Negated** Driven by the Bus Master to indicate that the current datum can complete.
CSTOP#	I/O	**Stop Negated** Driven by the Bus Slave 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.

Table 2 CardBus Pin Description (Page 1 of 2)

Notes

Name	Type	Description
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.
CBLOCK#	Input	**Lock Negated** Driven by the Bus Master to indicate that an exclusive operation is occurring.
CCD[2:1]#	Output	**Card Detection Negated** Indicate insertion and removal of the CardBus device. Require external pull-up resistor.
CVS[2:1]	I/O	**Voltage Sense** Indicate 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 2.1 specification. To be compatible with the RC3233x 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

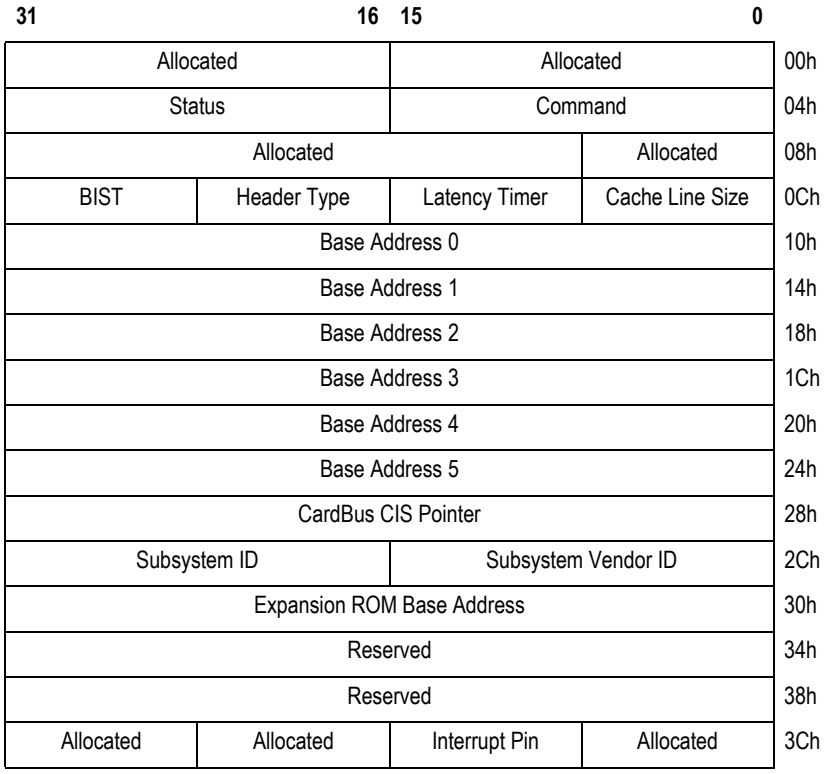

Figure 2 CardBus Configuration Header Space

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 process an idsel 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 that of 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 2.1 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 RC3233x 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 idsel 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 RC3233x GPIO pins, which acts as an idsel and is used to control the assertion of frame_n 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.

CardBus facilitates a single interrupt line, CINT#, unlike PCI which incorporates four interrupt signals, INT[D:A]#. Applications using more than one PCI interrupt signal must OR the PCI interrupt signals to the CINT# signal.

The Hot Insertion/Removal capabilities of the CardBus device are not supported by the RC3233x. 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 RC3233x. Therefore, the CCLKRUN# signal is tied to ground to disable this option.

Figure 3 shows the RC32334 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 RC3233x in the same manner, excluding the Bus Request and Bus Grant signals that the Target CardBus device would not have.

Notes

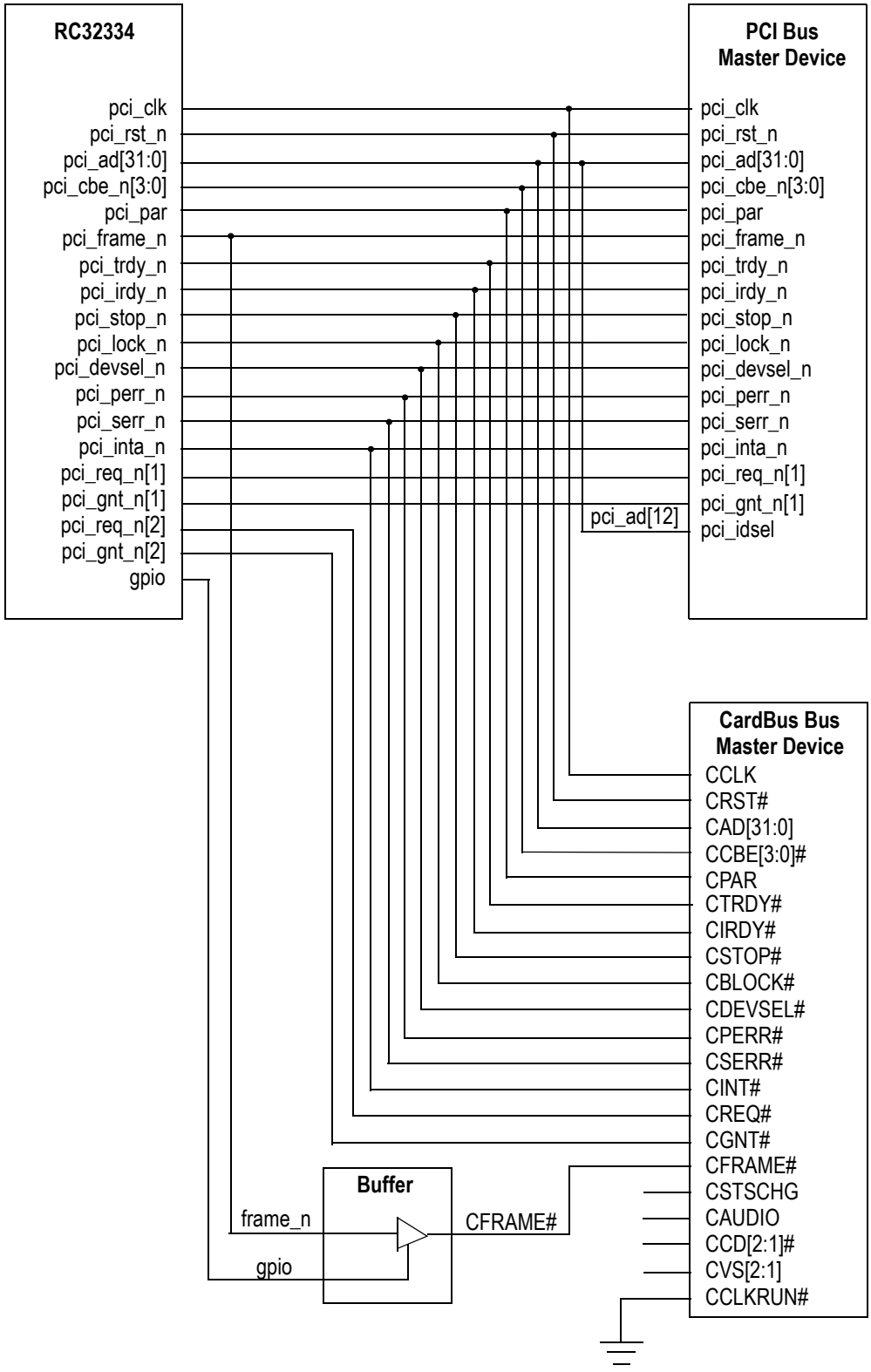

Figure 3 Connection Between the RC32334, a PCI Bus Master Device, and a CardBus Bus Master Device

Notes

Conclusion

The PCI and CardBus protocols are very similar, but they do differ in 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 RC3233x processors without implementing a PCI to CardBus Bridge device in the system, thus reducing design time, board cost, and board real estate.

References

IDT79RC3233x Integrated Communications Processor User Reference Manual.

PC Card Standard specification.

PCI Revision 2.1 specification.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

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

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.