Application Note

PowerPC Embedded Processors

Setting Up the CPC700 PCI Configuation Registers

Abstract - This note describes how to set up the CPC700 Memory Controller's PCI configuration registers. The information highlights and complements the information found in the *CPC700 Functional Specification*, the *CPC700 Memory Controller and PCI Bridge Datasheet,* the *PCI Bridge Technical Datasheet,* and the associated hardware specifications.

General

The CPC700 PCI interface can be used in either host mode or adapter mode. The CPC700 supports both synchronous and asynchronous PCI operation. In synchronous mode, the PCI interface bus is run at 1/2 the system clock. Asynchronous mode is used for PCI bus speeds higher than 33 MHz[1].

As shown in *Table 1*, the CPC700 can do one of three things with respect to configuration cycles originating from the processor local bus (PLB).

Table 1.

Action	Bus Number	Device Number
Access internal configuration registers	0	0
Execute a type 0 configuration cycle on the PCI bus[1]. Type 0 configuration access is used to configure a device on the PCI bus on which the configuration is run.	0	non-zero
Execute a type 1 configuration cycle on the PCI bus[2]. Type 1 accesses are used to configure a device on a lower-level PCI bus in a system with hierarchical PCI buses.	non-zero	any number
Restrictions (apply whether the CPC700 is an adapter or a host): 1. Cannot perform type 0 cycles to device number 0. 2. Cannot perform type 1 cycles to bus number 0.		

As either a host or an adapter, the CPC700 can execute type 0 or type 1 cycles with the above restrictions as long as the restrictions do not violate the system's requirements.

Even though the internal configuration accesses require a value of zero in the bus number field of the configuration address, as either a host or an adapter the CPC700 can reside on a bus other than zero and accept configuration cycles. These internal accesses are independent of the actual bus number to which the CPC700 is attached.

PCI Arbiter

The CPC700 includes an internal PCI arbiter that supports up to six external PCI devices at 33 MHz and can support one internal PCI master engine at 66 MHz.

When using the internal arbiter, it is not necessary to connect any CPC700 pins back around to an arbiter input. The arbiter is enabled and disabled based on the boot strapping of the CPC700 TT[4] signal. The CPC700 has a series of strapping pins that are used during power-up to select various configuration parameters such as PCI frequency, internal PCI arbiter, data parity, and several other parameters. The state of the strapping pins is captured in the CPRSTRAPREAD register and can be read by software.

The CPC700 PCI arbiter uses a fixed, balanced-tree priority scheme. Input 0 of each individual block has priority over input 1 of the block. This means request 0 has the highest priority while request 5 has the lowest. Bus parking modes and maximum transfer counts per-grant can be programmed via the PCI arbiter control register.

PCI Address Map Initialization

The local configuration registers control the processor local bus (PLB) to PCI address-translation mechanism, which includes PCI master mapping (PMM) and PCI target mapping (PTM).

PCI master mapping allows three regions of the PLB address space to be mapped into the PCI address space when the CPC700 is a PCI master. For example, PLB address x'8000 0000' can be mapped to x'0000 0000' on the PCI bus.

[1] In asynchronous mode, a PCI bus running at 33 MHz will have lower performance (more latency and less bandwidth) than a PCI bus running at 33 MHz in synchronous mode.

PCI target mapping allows a PCI address to be mapped to a PLB address when the CPC700 is a PCI target. For example, PCI address x‘0000 0000’ can be mapped to x‘0000 A000’.

CPC700 Bridge as Master

When the CPC700 bridge is the master on the PCI bus, it can generate memory, I/O, configuration, and special cycles. The CPC700 address sizes used to generate these cycles are all fixed, except for memory cycles.

The PCI master mapping (PMM) registers specify the PCI memory-cycle sizes and the PLB address space. In addition, the PMM registers specify the translation that enables the address of the resulting PCI memory cycle to be offset from the PLB address. The PMM registers do not default to usable values following reset; they must be initialized before attempting to generate PCI memory cycles.

CPC700 Bridge as Target

When the CPC700 bridge is a target on the PCI bus, it can respond to memory cycles. To set up the PCI target address map:

1. Write valid values to the PCI target mapping (PTM) local address registers to control the translated PLB address. The PTM local address registers do not default to usable values following reset; they must be initialized before responding as a PCI memory target is possible.
2. Write valid values to the PTM size and attribute registers to control the behavior of the base address registers (BAR1 and BAR2).
3. Set the host configuration enable bit (bit 0) of the bridge options 2 register.

   Note: If the host configuration enable bit is cleared, the PCI target will retry the configuration cycles. The goal is to prevent the PCI configuration software from doing anything with the BARs (via configuration cycles) before PTM 1 and 2 have been properly set up.
4. The BARs are typically initialized as part of the standard PCI initialization process. BAR1 and BAR2 specify the memory-cycle address ranges to which the CPC700 responds. BAR1 must be used since its enable is hard wired active. If BAR2 is to be used, then write the PTM 2 enable bit and the PTM 2 registers normally (prior to setting the host configuration enable bit).
5. Set bit 1 of the PCI command register, which controls memory accesses to the PCI target. This bit must be set for the CPC700 to respond to PCI memory cycles. Otherwise, it will ignore them (resulting in a master abort on the bus). This bit is disabled at reset. If this bit is set before performing the steps described above, the outcome is unpredictable.

Other Registers That Must Be Initialized

Error handling is initially disabled (error detection masked). If error handling is to be enabled, then the error enable register must be initialized appropriately.

The bridge options 1 and bridge options 2 registers contain a variety of options that may need to be changed.

Adapter Mode Configuration Access

When the CPC700 is used as a PCI adapter interface (in adapter mode), it can respond as a configuration target. The PCI_IDSEL pin must be connected and active for the CPC700 to respond as a configuration target. Before the host runs the PCI configuration sequence, the following registers must be appropriately initialized by the local CPU with values taken from a local ROM:

- PCI Vendor ID
- PCI Device ID
- PCI Revision ID
- PCI Class
- PCI Subsystem ID
- PCI Subsystem Vendor ID

Accessing PCI Registers

The following must happen before a PCI bus master can successfully access the bridge PCI port:

1. Configure the memory map by initializing the PTM 1 register, or the PTM 2 register, or both. (These are local configuration registers and must be configured from the PowerPC® 6xx/7xx processor bus.)
2. Set the memory access bit (bit 1) of the PCI command register (offset x‘05’ to x‘04’) to enable the bridge as a target for PCI memory cycles.
3. If the internal PCI arbiter (PCI arbiter control register at offset x‘47’ to x‘44’) is used, it may require configuration. The arbiter is enabled or disabled by strapping pin TT[4] as shown in the general strapping options table in the *CPC700 Memory Controller and PCI Bridge User’s Manual*.
4. Set the host configuration enable bit (described on page 2) to allow configuration from the host side (PCI bus master).

After these steps are performed, a PCI bus master can access the bridge PCI port. The PTM BARs should be configured first (as in any standard PCI configuration sequence).

5. A PCI bus master has access to all of the internal configuration registers except the local configuration registers (that is, the registers from x‘FF40 0000’ through x‘FF40 00FF’), CONFIG_ADDRESS, and CONFIG_DATA. The *CPC700 Memory Controller and PCI Bridge User’s Manual* lists the exact form of access (read and write or read only) for each register. Some of these registers allow reading and writing from the local (PLB) side, but only reading from the PCI side (for example, the PCI vendor ID register at offset x‘01’ to x‘00’).
6. To enable a PCI bus master to access configuration registers, first set the host configuration enable bit (bit 0) of the bridge options 2 register (at offset x‘61’ to x’60’). If this bit is not set, the PCI bridge will retry all attempts by a PCI bus master to perform configuration cycles to the bridge.

Setting up PCI Configuration

1. Scan the PCI bus for PCI devices or PCI-to-PCI bridges.
2. Determine the memory and I/O requirements of each device by reading its base address registers (using PCI configuration cycles).
3. Assign each device a unique range of PCI addresses by writing the starting address of the range back into the base address registers. The device will use this value to decode the normal PCI memory and I/O cycle addresses.

Device Driver

1. The device driver is pre-programmed with the vendor ID and device ID of the device it supports.
2. It calls a function that scans the bus for a match and returns the device number.
3. The device driver then uses PCI configuration cycles to determine the PCI address ranges that were assigned to the device by the software (BIOS). It completes all other necessary PCI initialization of the device.
4. The device driver uses PCI memory or I/O cycles or both to do device-specific initialization; it can then enable the device.

Terms

base address registers (BAR). Device configuration registers that define the start address, length, and type of memory space required by a device. The type of space required will be either memory or I/O. The value written to this register during device configuration will program its memory or I/O address decoder to detect access within the indicated range.

bus master. A device capable of initiating a data transfer with another device.

processor local bus (PLB). The PLB is a high performance on-chip bus; it supports read and write data transfers between master and slave devices equipped with a PLB interface and connected through PLB signals.

type 0 configuration access. Type 0 configuration access is used to configure a device on the PCI bus on which the configuration is run.

type 1 configuration access. Type 1 accesses are used to configure a device on a lower-level PCI bus in a system with hierarchical PCI buses.

Revision Log

Revision Date	Modification
December 13, 2000	Initial release (00).
March 28, 2002	Second release (01). Reformatted to match new template. Revised and rewritten to improve clarity.

Printed in the United States of America March 2002.

All information contained in this document is subject to change without notice. The products described in this document are NOT intended for use in implantation or other life support applications where malfunction may result in injury or death to persons. The information contained in this document does not affect or change IBM product specifications or warranties. Nothing in this document shall operate as an express or implied license or indemnity under the intellectual property rights of IBM or third parties. All information contained in this document was obtained in specific environments, and is presented as an illustration. The results obtained in other operating environments may vary.

THE INFORMATION CONTAINED IN THIS DOCUMENT IS PROVIDED ON AN "AS IS" BASIS. In no event will IBM be liable for damages arising directly or indirectly from any use of the information contained in this document.

IBM Microelectronics Division
1580 Route 52, Bldg. 504
Hopewell Junction, NY 12533-6351

The IBM home page can be found at
http://www.ibm.com

The IBM Microelectronics Division home page can be found at
http://www.chips.ibm.com

IBM Microelectronics
4400 Silicon Drive
Research Triangle Park, NC 27709
ppcsupp@us.ibm.com
(919) 543-5701

CPC700Regset.fm.00
March 28, 2002 IBM Confidential