RIOC 4068

PowerPC-Based CompactPCI Twin Bus Processor Board

User's Manual
DOC 4068/UM
Version 1.1
February 2004

CREATIVE ELECTRONIC SYSTEMS S.A.

CES Warranty Information

The information in this document has been checked carefully and is thought to be entirely reliable. However, no responsibility is assumed in case of inaccuracies. Furthermore, CES reserves the right to change any of the products described herein to improve reliability, function or design. CES neither assumes any liability arising out of the application or use of any product or circuit described herein nor conveys any license under its patent rights or the rights of others.

WARNING - FCC COMPLIANCE

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

CES - Creative Electronic Systems S.A.
38 Avenue Eugène-Lance
P.O. Box 584
1212 Grand-Lancy 1
Switzerland

Tel: (41) 22.884.51.00
Fax: (41) 22.794.74.30
EMail: ces@ces.ch

February 24, 2004
A. Martin

Revision History

RIOC 4068 PowerPC-Based CPCI Twin Bus Processor Board

Version Number	*Modifications*
version 1.1	- Correction Chapter 9 : tables 9-5 and 9-11 - Preface added and relative modifications

Table of Contents

RIOC 4068 PowerPC-Based CPCI Twin Bus Processor Board

List of Tables

List of Figures

RIOC 4068 PowerPC-Based CPCI Twin Bus Processor Board

Preface
Unpacking and Installing your CES Boards

A. General Precautions

ESD PREVENTIVE MEASURES

- Ensure that you and all of the electrical equipment that you handle during this installation are properly grounded to avoid damage from ESD. The best way is to attach a ground strap to your wrist when handling any boards or peripherals.
- Keep boards in their antistatic packing until they are needed; remove a board from its antistatic bag only when you are about to install it.
- Place the boards only on an antistatic mat when manipulating them (setting switches, adding-boards etc.). Do not place the boards on top of an antistatic bag unless the outside of the bag also has antistatic protection.

MECHANICAL PREVENTIVE MEASURES

- Protect your board from mechanical shocks when inserting and removing it into/from the chassis.
- Pay careful attention to the components on the solder-side of the boards when inserting them into or removing them from the chassis.

ELECTRICAL SAFETY MEASURES

- Please follow all appropriate safety regulations and practices when installing equipment.

B. Unpacking

Your board is delivered in a box labelled with the name, reference number and serial number of the board; check that they correspond to your order before you open the box!

The board itself is packed in special antistatic materials within the box; when you unpack the board, save the packaging material; you may need it to transport the board at a later date.

Before you open the box, make sure that both you and your environment are properly protected against ESD.

C. Installation

WARNING This equipment must be handled with care!

- Make sure that the ESD protection for you and your environment is fully operational before removing the board from its plastic bag.
- If you need to set any configuration switches or jumpers on the board, refer to the User's Manual for their correct configuration. Then place the board on an anti-static surface which is properly grounded before setting them.
- If you are attaching your card to a motherboard, then first place that board on an anti-static surface. Then plug your card in place, and insert and tighten the fixing-screws carefully.
- Unless CES has confirmed that the card may be inserted into a crate with power applied, make sure that power to the crate is disconnected.
- Plug the board carefully into the chassis; do not use excessive force. Beware of physical contact between the card you are inserting and the neighbouring cards, particularly if the operation is carried out while the crate is powered-up.
- Pay particular attention to the ESD gasket and backplane connector while manipulating boards under tension.
- Attach any front-panel connections.
- Apply power to the crate.

D. If you have problems...

...either during first installation of later...

The first thing to do is to contact the CES support hotline or send an e-mail to

hotline@ces.ch

describing the symptoms and giving type and serial no. of the card together with details of how you can be contacted.

A member of the support team will contact you to help you, and to determine what action to take.

Should you be advised to return the board to CES, please use the original packing material if possible.

Complete the RMA form supplied with your board in the original package and enclose a copy of it with your board.

Follow the instructions on the RMA form for returning the board to CES.

E. Environmental Specifications

Standard

Temperature	Operating Storage	0°C to 50°C with forced-air cooling (400 LFM) -40°C to 85°C
Humidity	Relative Humidity	5% to 85% non-condensing

REGULATORY COMPLIANCE

Immunity	EN 50082-2 / EN 55024
Emission	EN 55022-A
Safety	EN 60950

WARNING CES accepts no responsibility for equipment failure, should these recommended precautions not be respected.

Chapter 1 Getting Started

1.1 General Description

The RIOC has been designed to provide a high-end CompactPCI platform for multi-processor-based applications, which require the maximum available speed on PCI and CompactPCI simultaneously from multiple sources.

Its design characteristics have been aimed at the maximum throughput on each of the available buses. The RIOC 4068 provides a function-level compatibility with the RIOC 4065. This third generation of CES PowerPC cards is built around FPGA-based CES-designed PowerPC-to-PCI and PowerPC-to-CompactPCI bridges to reduce the risk of long term maintenance and to eliminate the limitations of those available on the market.

THE RIOC OFFERS A UNIQUE SET OF CHARACTERISTICS:

- Three bus architecture:
  - PCI#1: local PCI
  - PCI#2: J3 PCI bus
  - PCI#3: CompactPCI 3.3 and 5 Volts Siganling
- Hardware-designed transparent multiprocessor support
- Two onboard PMC slots

THE RIOC OFFERS UNMATCHED TUNING CAPACITIES:

- Versatile interrupt strategies: guaranteeing an interrupt service within 5 µs independent of the other pending requests. It is made possible under OS due to the unique CES-designed interrupt controller.
- Hardware chained block transfers: sustained 80 / 160 MBytes/s on CompactPCI are achieved without any impact on the PowerPC performance.

1.2 Features

BRUTE FORCE POWER

PowerPC 7455 (G4+) at maximum available frequency (for Low power spec.).
2 MBytes MSUG2 DDR SRAM Backside L3 Cache up to 200 Mhz

GLOBAL MEMORY AT CACHE SPEED

Up to 1 GByte SDRAM at 800 MBytes/s peak.

TWIN 64-BIT PCI BUS ARCHITECTURE

- Local PCI mezzanine slots (2 PMCs)
- PCI Rear Panel extension
- Transparent multi-processing over PCI and CompactPCI.

HIGH-PERFORMANCE COMPACTPCI INTERFACE

- Automatic Selection of CompactPCI System Slot or Peripheral Slot
- Hot-Swap. Direct memory-to-CompactPCI interface. Multiple independent DMA channels on CompactPCI at full CompactPCI speed.

TOTALLY UNDER CONTROL

Multiple arbiters for memory bandwidth allocation. User-controlled interrupt strategies. Network protocols on PCIs and CompactPCI (TCP/IP). JTAG 1149-1 support.

TOTALLY SCALABLE

PCI 64-bit based on the J3 backplane connector. Up to six PMCs controlled by the same RIOC. Hot-Swap compliant.

NATIVELY MULTIPROCESSOR ORIENTED

Unique multiprocessor clustering which features:

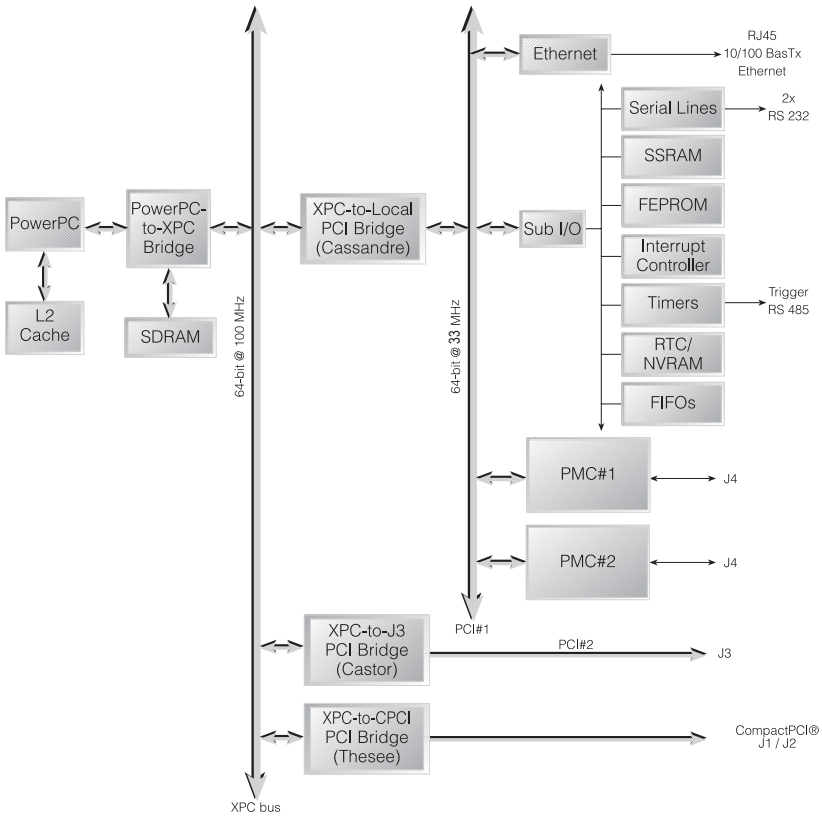
- High-speed synchronization
- High-speed message passing
- Global shared memory

STATE-OF-THE-ART SOFTWARE

Extended Board Support Packages, System Support Packages and services are available for VxWorks, LynxOS and Linux including all of the major telecommunication and aerospace drivers (ATM, AFDX, 1553B, etc...).

1.3 Block Diagram

Figure 1.1 RIOC 4068 Block Diagram



1.4 Specifications

CPU SUBSYSTEM

CPU Type	Low Power (Spec N) PowerPC 7455 (G4+) from 600 MHz
Global Memory	64 / 128 / 256 / 512 MBytes or 1 GByte SDRAM at 100 MHz G4: 32 KBytes data / 32 KBytes instruction L1 - 256 KBytes L2 - 2 MBytes backside L3 cache
Flash EPROM	16 MBytes, up to 48 MBytes with software compression
Timers	One TICK Timer and six user-controlled timers
RTC/NVRAM	M48T59 chip with replaceable battery backup, 8 KBytes NVRAM, Watchdog Timer, Real-Time Clock

PCI BUSES

PCI Bridges: CES-designed for performance optimization
- PPC-to-PCI#1, 64-bit at 33 MHz
- PPC-to-PCI#2 , 64-bit at 33 MHz
- PPC-to-64-bit CompactPCI at 33 MHz
- Compliant with PCI revision 2.2

PCI Mezzanine Card (PMC)
- Two IEEE P1386 compliant onboard PMC slots with front panel and backplane I/O (J4)
- Only PMC#2 is 5 Volts tolerant
- Hardware provision made to support future extensions of the PPMC standard

PCI Extension Boards (PEB 6418)
- Two additional PMC slots per PEB, up to two PEBs attached to the same RIOC

COMPACTPCI INTERFACE

- 64-bit at 33 MHz
- Automatic selection of System & Peripheral slot functionalities
- 2 GBytes CompactPCI master addressing space (128 user-programmable windows of 16 MBytes)
- CompactPCI Slave
  - 32-bit: 512 MBytes CompactPCI slave addressing space (512 user-programmable windows of 1 MByte)
- 16 independent DMA channels with dedicated FIFOs
- Compliant with PICMG Hot-Swap specification, PICMG 2.1 revision 1.0

FRONT PANEL I/OS

Serial Ports	Two ports. Up to 135 Kbits/s (synch.), 38.4 Kbits/s (asynch.)
Ethernet	PCNET 79C973 controller 10 Base-T / 100 Base-Tx (auto-negotiate speed select) with 32-bit PCI DMA - RJ45 connector

J4 BACK I/OS

PMC's	Two time 32 bits User defined I/0 connected to the PMC's Pn4 connectors
User	General purpose I/O (remote reset, RS485 trigger I/O, etc...)

HARDWARE MULTIPROCESSOR SUPPORT

- Read-modify-write decoding
- Multi-port global memory
- Autonomous DMA logic
- Eight 255 x 32-bit FIFOs with interrupt capability
- Complete interrupt structure

MECHANICAL SPECIFICATIONS

PCI Mezzanine	Two PMC slot extensions
CompactPCI Module	Single-slot 6U (233.4 mm x 160.0 mm)

POWER DISSIPATION

+3.3 V	2.5 A
+ 5 V	6 A
± 12 V	10 mA

NOTE

For environmental specifications and regulatory compliance, please refer to the Preface.

Chapter 2
Installation Notes

2.1 General Precautions

Please read carefully the Preface.

2.2 Board and Layout Description

Figures 2.1 and 2.2 show the layout of the RIOC 4068.

The main components are:

- PowerPC 7455 (Low Power Spec.), running at the maximum available speed.
- The system SDRAM, residing on a daughter-board.
- The PowerPC-to-XPC bus bridge. This CES-designed FPGA connects the XPC bus (used as the backbone of the board) to the PowerPC.
- THESEE, the XPC-to-CompactPCI bridge. This CES-designed FPGA creates the interface between the XPC bus and the CompactPCI bus.
- CASSANDRE, the XPC-to-local PCI bridge. This CES-designed FPGA connects all onboard PCI resources of the board (Ethernet, SubI/O, etc.) to the XPC bus.
- CASTOR, the XPC-to-J3 bridge. This CES-designed FPGA allows the user to control up to four additional PMCs mounted on PEBs (PCI Extension Boards) and allows them to share a common PCI bus routed to the J3 connector.
- The SubI/O interface. This CES-designed FPGA interfaces all non-PCI resources of the board (SRAM, RTC, NVRAM, SIC, etc.) to the local PCI bus. It also contains seven timers.
- The Flash EPROM. This 16-MByte device contains boot-up code and PPCMon and can be loaded with romable kernels and user applications.
- The SRAM. This 512-KByte memory is used as FIFO storage and user scratchpad memory.
- The RTC and NVRAM. This device provides a Real-Time Clock on the RIOC 4068 and an 8-KByte non-volatile RAM for storage of the boot parameters.
- The System Interrupt Controller.
- A serial EPROM, which stores the configuration data for all reprogrammable logics of the board.

Figure 2.1 RIOC 4068 Board Layout (Component Side)

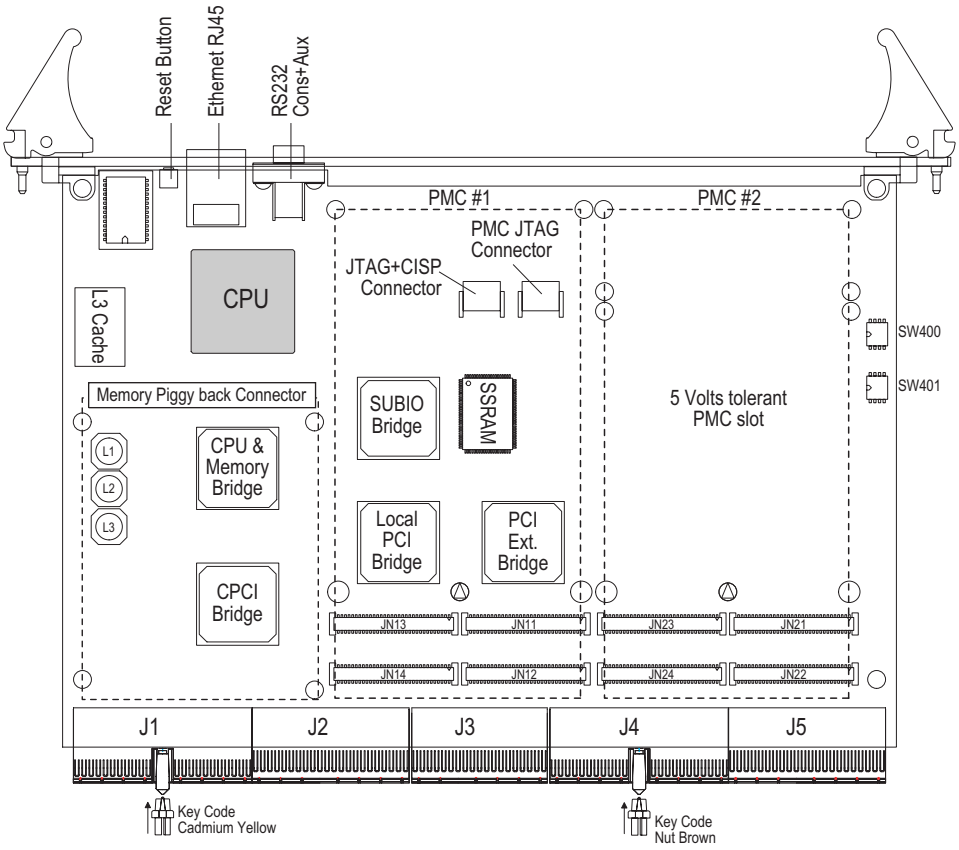
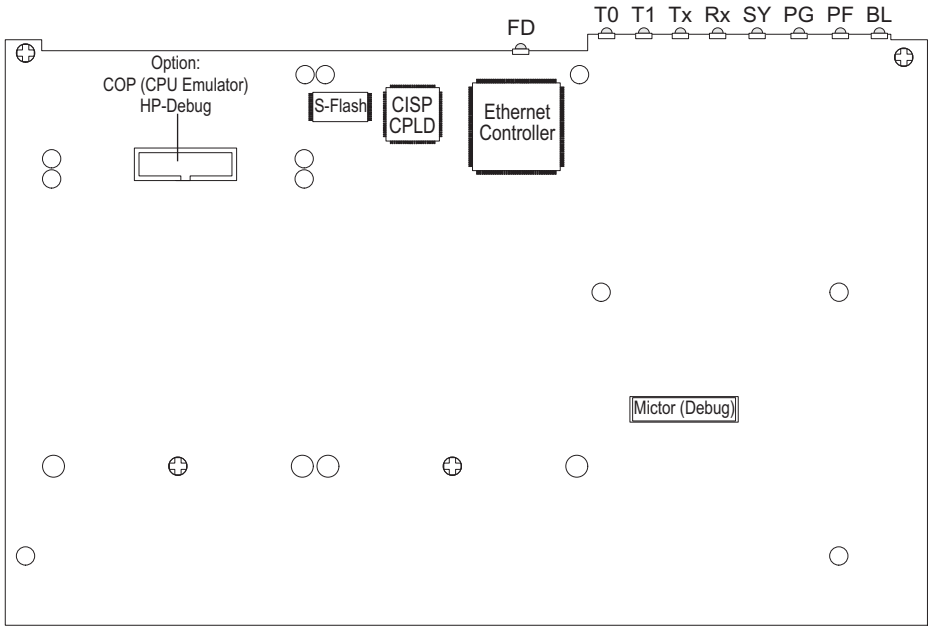


Figure 2.2 RIOC 4068 Board Layout (Solder Side)



2.3 Jumper and Switch Setting

2.3.1 SW400 - PCI Extension Bus Setting

This four-position SMD DIP switch allows to set the parameters of the PCI bus routed to the J3 connector as the PCI bus extension.

Table 2-1 SW400 - PCI Extension Bus Setting

Position	*Function Name*	*Description*	*Factory Setting*
SW400-1	Spread-CLOCK	ON: select the 100 MHz for Spread Clocking (0.1 MHz step according the RIOC geographic address)	OFF
SW400-2	J3_CLOCK	ON: select dual clock for new BPA 6414B OFF: select single clock for BPA 6414A	OFF
SW400-3	RESERVED	Reserved	OFF
SW400-4	PMC_JTAG	Shortcuts TDI/TDO on the PMC#1 for single PMC JTAG programming (CES reserved)	OFF

WARNING SW400-4 should be OFF when a PMC is plugged into the bottom PMC slot.

2.3.2 SW401 - System Set-Up

This four-position SMD DIP switch is used to configure the reloading of the onboard reprogrammable logic, in order to try to boot in an emergency boot mode in case the Flash EPROM has accidentally been erased. It also selects the type of local reset (cold or warm) generated by a CompactPCI reset.

Table 2-2 SW401 - System Set-Up

Position	*Function Name*	*Description*	*Factory Setting*
SW401-1	CPLD_PROG	ON: selects short JTAG chain for CPLD programming OFF: selects full JTAG chain for BSCAN test	OFF
SW401-2	PPC_DEBUG	ON: selects HP_PROBE JTAG for CPU debugger OFF: selects JTAG_BSCAN test for CPU	OFF
SW401-3	BOOT_SEL	ON: selects primary boot on serial Flash OFF: selects boot on Flash EPROM	OFF
SW401-4	CPCI_RxRESET	ON: CPCI reset generates a cold onboard reset OFF: CPCI reset generates a warm onboard reset	OFF

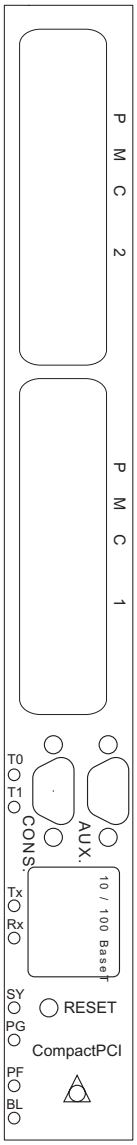
NOTE SW401-1 should be ON only for CPLD programming (CISP tool).
SW401-2 should be ON only when used with a special HP-COP debugging tool.
SW401-3 should be ON to restart the RIOC when standard Flash EPROM has been erased.

2.4 Front Panel Description

The front panel implements the following devices:

- Reset button
- RJ45 ethernet
- 2 x RS232 console / auxiliary
- LEDs display

Figure 2.3 RIOC 4068 Front Panel



2.4.1 Reset Button

The reset push button can be activated through the front panel by using a “specific nail tool”.

When pressed for more than two seconds, the reset logic will execute a “cold reset”, which will re-load all of the FPGA (LED “FD” light is OFF) and restart the CPU (hard reset).

When pressed for less than two seconds, the reset logic will execute a “warm reset”, which will re-initialize the logics (logic reset) and request a CPU exception (soft reset).

2.4.2 RJ45 Connector

The RJ45 connector integrates the coil and the “Bob-Smith Terminators” for the 10 Base-T / 100 Base-Tx ethernet controller and connects the UTP5 cable to the network.

2.4.3 RS232 Connectors

The front panel provides two RS232 full modem channel connections through µDB9 standard connectors. One is initialized by software for the console port and the other one for the auxiliary port.

2.4.4 LED Display

Table 2-3 LEDs

LED	*Function Name*	*Color*	*Description (when the LED light is ON)*
BL	HS_BLUE	Blue	When the Hot-Swap insertion / extraction protocol is in STATE_BLUE
PF	POWER_FAIL	Red	When the power supply controller detects an over-current condition
Tx	LAN_Tx	Green	When the ethernet controller transmits data
Rx	LAN_Rx	Green	When the ethernet controller receives data
SY	SYSTEM_SLOT	Orange	When the RIOC board is the system slot
T0	SUBIO_LED	Orange	When the PCI1_LOC_REG#6 bit [07] is set to 1
PG	PWR_GOOD	Green	When the RIOC power supplies are OK
FD	FPGA_DONE	Green	When all of the FPGAs are correctly downloaded
T1	LAN_STATUS	Orange	When the ethernet user-defined status is met (LAN speed, etc.)

2.4.5 Back I/O Connector

The CompactPCI J4 connector provides the optional connections for the following functions:
- Remote reset
- RS485 channels
- RS485 trigger
- PMC User's defined I/O (2 x 32 Bits)

REMOTE RESET (OPTO COUPLER)

A cold or warm reset (depending on the pulse length can be generated remotely from an external agent.
> 2 seconds = COLD RESET
< 2 seconds = WARM RESET

RS485 CHANNELS

These two Input / Output RS485 lines are fully managed by programming the PCI1_LOC_REG#0. These signals allow only to receive / transmit one-bit data at a time from / to an external agent.

RS485 TRIGGER

This Input / Output RS485 line is fully managed by programming the timer control register. This line should be used as a trigger input interrupt or to supply a trigger output pulse from the timer logic.

PMC's User I/O
These 2 x 32 bits Input / Output are connected to the PMC Jn14 and Jn24 Connector (User defined PMC I/O)

2.5 Installation Steps

Please follow the following steps for installation of a RIOC 4068 into a CompactPCI crate:
1. Be sure to be under ESD protection, as described in section 2.1.
2. Check the CompactPCI Chassis for 3.3 Volts Signaling (Yellow Key in the J1 connector)
3. Unpack the RIOC 4068.
4. Set the DIP switches located on the component side of the printed circuit of the RIOC 4068.
5. Eventually install your PMC mezzanines.
6. Plug the RIOC 4068 in a CompactPCI crate: carefully insert the RIOC board in the CompactPCI crate.
7. Connect the terminal.
- Connect a terminal to the serial line µDB9 connector labelled "Cons." on the front panel with a RSC 6731A0 cable.
- Set your terminal operating mode to:
  - Tx baud rate 9600
  - Rx baud rate 9600
  - 8-bit no parity

- One-stop bit
- Local echo OFF
- Newline OFF
- del=H

8. Connect the RIOC 4068 on a network.

- Connect the RJ45 cable (ECC 6714A0) from the RIOC 4068 to a concentrator box. The RIOCs support both 10 Base-T and 100 Base-Tx connections with auto-negotiation.

9. Power ON the system.

- Power ON first the peripheral devices (terminal, SCSI devices).
- Power ON the CompactPCI crate.

NOTE

The RIOC 4068 supports the Hot-Swap feature, and thus can be inserted / extracted in / from a powered CompactPCI crate.

2.6 RIOC Boot Sequence

The boot sequence of the RIOC 4068 is controlled by the micro-switch SW401-3. This switch should be OFF for a normal mode of operation of the board. It can be switched ON in case of trouble, as described in section 2.7.

At power ON, or after a reset, the onboard PPCMon firmware of the board initializes the resources of the RIOC 4068 (CPU, FPGA initialization, device check, etc.) and shows the result of the self-test routines executed (`OK`, `not OK`, `skipped`). The PPCMon prompt is displayed at the end of this initialization and diagnostic sequence.

```
MINI_Mon:booting PPC_Mon

PPC Boot Rev 4.8 created Fri Jun 23 16:27:02 2000

  Module Type: 4068BE  rev: 00  serial: 109
  Host CPU: PPC7455   ver: 83.02   speed: 800 Mhz
  PCI Bridge: CES XPC
  Memory Size: 128 + 128 = 256 Mbytes
  L3 CACHE:  2 Mbyte DDRAM
  FPROM: Two banks of AMD29F032 installed [16 Mbytes]
  Ethernet Address: 00:a2:80:01:48:cd

Entering boot diagnostics

 0    Check System Memory (0x00004000 - 0x0ff00000)          0  OK
 1    Check Interrupt Handler                                0  OK
 2    Check PCI Bridge                                       0  OK
 3    Check Cache and MMU                                    0  OK
 4    Check MK48T08 RTC and NVRAM                            0  OK
 5    Check SIC6351 Interrupt Controller                     0  OK
 6    Check Serial Lines                                     0  OK
 7    Check Micro Timers                                     0  OK
 8    Check FIFO's (0 - 7)                                   0  OK
 9    Check Digital Thermometers                             0  OK
10    Check PCI devices                                      0  OK
11    Check On Board Ethernet Controller  (PCI slot 0)       0  OK
12    Check PMC #1 Extension  (PCI slot 2)                   0  NO DEVICE
13    Check PMC #2 Extension  (PCI slot 1)                   0  NO DEVICE
14    Check CPCI Interface                                   0  OK

***********************************************
*   PPC_Mon RIOC4065 monitor - version 4.8       *
*      CES SA Copyright 1995 - 2000              *
***********************************************

PPC_Mon>
```

NOTE

- Please refer to the PPCMon User's Guide for instructions regarding the use of PPCMon.
- Pressing "x" for several seconds, allows PPCMon to execute the initialization and diagnostic sequence and gives the control back to the user (when an auto-boot of an OS has been selected).
- Pressing " " (space bar) stops the PPCMon initialization and diagnostic sequence and immediately gives the control back to the user. In this case, some resources of the board may not be initialized.

2.7 Trouble-Shooting Tips

This section describes the main problems, which can prevent the user from operating the RIOC 4068 successfully.

2.7.1 Unable to Insert the Board into the Backplane

- Verify that the CompactPCI Chassis implements 3.3 Volts V_I/O (Yellow Key in J1 connector).
- Verify that there are no mechanical and physical obstructions in the CompactPCI slot.
- Verify that the pins of the CompactPCI backplane connectors are not bent or damaged.

2.7.2 Unable to Get the PPCMon Prompt

PG LED ON THE FRONT PANEL IS OFF

The chassis power supplies do not comply with the specifications. Verify the power supplies (3.3 V, + 5 V, + 12 V, - 12 V) of your CompactPCI crate.

PF LED ON THE PRINTED CIRCUIT IS ON

The power supply controller has detected an over-current condition onto the RIOC board. Please contact CES support.

FD LED ON THE PRINTED CIRCUIT IS OFF

The FPGAs of the board could not be loaded. This might indicate a hardware problem. Please contact CES technical support.

NOTE

Refer to Figure 2.2 for LEDs location.

NOTHING IS DISPLAYED ON THE CONSOLE

- Verify your RS232 cable.
- Verify that the RS232 cable is connected to the µDB9 connector labelled "Cons.".
- Verify that you are not using a "null-modem" cable (pins 2-3 crossed).
- Verify that your terminal is set according to the instructions given in section 2.5.

PPCMON HAS ACCIDENTALLY BEEN ERASED

- Be sure to be under ESD protection.
- Remove the board from the crate and switch SW401-3 to ON.
- Insert the board into the crate and power ON the system.
- You should enter a small monitor and get a MiniMon> prompt.
- If this is not the case, contact CES technical support.
- Type "q" and "return", you should enter a backup copy of a PPCMon firmware.
- Reload PPCMon according to the procedure described in the PPCMon User's Guide.
- Power OFF the crate, remove the board and switch SW401-3 to OFF.
- Insert the board into the crate and power ON the system.
- If you do not get the PPCMon> prompt, contact CES technical support.

PPCMON SHOWS A NOT OK STATUS DURING THE BOOT DIAGNOSTIC

Make a copy of your terminal screen and contact CES technical support.

PPCMon detects a NVRAM Checksum Error

The NVRAM contains vital parameters of the board. This NVRAM is protected by a checksum. In case of corruption of the NVRAM, PPCMon displays an error message and waits for a user-action within the next five seconds.

If the user types any characters during this time, PPCMon lets him re-enter all of the parameters of the board.

If the five seconds of time-out expires, PPCMon searches if a “PPCMon> nvram save” has already been executed. If this is the case, NVRAM parameters are extracted from the backup copy of the NVRAM in the Flash EPROM and resets the board.

If no backup copy is available, the user is invited to re-enter all of the parameters of the board.

PPCMon does not give Back Control to the User after its Initialization and Diagnostic Process

- You probably have configured PPCMon with a “PPCMon> setenv boot” value equal to 50. Set this value to 0.
- Press “x” on the keyboard for several seconds to get control after the end of the PPCMon start-up process.

Chapter 3
Computing Core

3.1 CPU

The RIOC 4068 board provides the footprint for the PowerPC 7455 (G4+) processor along with its associated back-side L3-cache. The speed grade depends on the version and the avaibility of the G4+ in the low power specification.

3.1.1 CPU Physical Address Space

On the RIOC 4068, the PowerPC address space is divided between the DRAM, Local_PCI, PCI_J3 and CompactPCI.

Table 3-1 CPU Physical Address Space

CPU	*Port Address*	*Size*
0x0000'0000 - 0x3FFF'FFFF	System Memory	1 GByte
0x4000'0000 - 0x7FFF'FFFF	CompactPCI	1 GByte
0x8000'0000 - 0xBFFF'FFFF	PCI _J3 or CompactPCI	1 GByte
0xC000'0000 - 0xFFFF'FFFF	Local_PCI	1 GByte

3.2 Main Memory

The RIOC 4068 comes in configurations of 64 MBytes, 128 MBytes, 256 MBytes, 512 MBytes or 1 GByte. The system memory is controlled (refresh modes, bursts, etc.) by a CES-designed FPGA. The system memory is built with SDRAM clocked at 100 MHz and implements byte parity protection.

The SDRAM controller is optimized for the PowerPC G4, which gives high bandwidth memory access. Three different sizes of SDRAM components are used: 64 Mbits, 128 Mbits or 256 Mbits ones.

The SDRAM system memory is implemented on a small electronic card plugged on the motherboard (RIOC 4068). This small card is named a piggy card.

Table 3-2 Memory Address Space

CPU or PCI_{MEM}	*Board's Memory Size*	*SDRAM Technology*
0x0000'0000 - 0x03FF'FFFF	64 MBytes	64 Mbits
0x0000'0000 - 0x07FF'FFFF	128 MBytes	128 Mbits
0x0000'0000 - 0x0FFF'FFFF	256 MBytes	128 Mbits

Table 3-2 Memory Address Space

CPU or PCI_{MEM}	*Board's Memory Size*	*SDRAM Technology*
0x0000'0000 - 0x1FFF'FFFF	512 MBytes	256 Mbits
0x0000'0000 - 0x3FFF'FFFF	1 GByte	256 Mbits

NOTE The FPGA, which implements the SDRAM controller is specific to each SDRAM technology. Therefore, the memory upgrade could also require an FPGA code upgrade.

3.3 L1 Cache

The PowerPC 7455 (G4+) processor implements separate 32Kbytes eight-way set associative Level 1 (L1) Instruction and Data cache running at CPU core speed.

3.4 L2 Cache

The PowerPC 7455 (G4+) processor implements a 256KBytes eight-way set associative Level 2 (L2) unified Instruction and Data cache running at CPU core speed.

3.5 L3 Cache

The RIOC 4068 board implements 2 MBytes of MSUG2 DDRSRAM L3 cache.

Table 3-3 L3-Cache Devices

Late-Write SSRAM	*L3 Cache Size*	*Maximum L3 Clock Frequency (*)*
8 Mbits 3.3 ns	2 MByte	300 MHz

(*) **The L3 cache clock setting is selected by software through programming of the CPU registers and depends directly on the frequency of the CPU core.**

NOTE The L3 cache speed grade is dependent on the version.
For more information about the L1, L2 and L3 cache operations, please refer to the PowerPC 745x User's Manual.

The Motorola functional test operations, for the MPC7455, are tested for the L3 frequencies at maximum 200 MHz. (Refer to the Motorola Hardware Specification).

Chapter 4
XPC Bus

4.1 Overview

The XPC_bus is a 64-bit address / data multiplexed 100 MHz synchronous bus. It is implemented with low-level swing HSTL technology (1.5 [V]) to reduce the EMI / RFI. It is used to interconnect the four main onboard interfaces as follows:

- PowerPC-to-XPC bridge
- XPC-to-Local_PCI bridge
- XPC-to-J3_PCI bridge
- XPC-to-CompactPCI bridge

The XPC write transactions are always posted in write-buffer and are optimized for burst. Two different write categories are used to isolate the low-latency CPU writes from the high bandwidth DMA writes.

The XPC read transactions are fully compelled between the XPC master and the XPC slave. The XPC read request owns the highest priority over the XPC.

The IACK read transaction is used to read-out the vector from the System Interrupt Controller. A special flushing mechanism is associated with the IACK allowing to flush incoming write buffers on the specified XPC bridge.

4.1.1 XPC Bus Signals

The XPC bus is composed with the following signals:

Table 4-1 XPC Bus Signals

Signals	*Function*
xpcAD[63:00]	XPC address / data multiplexed
xpcADP[07:00]	XPC Byte parity (even) protection of xpcAD[63:00]
xpcMCOD[01:00]	XPC master to slave synchronization = 10 : transaction start = 01 : Intermediate state = 00 : transaction run = 11 : idle
xpcSCOD[02:00]	XPC slave to master synchronization = 000 : error + terminate the transaction = 001 : reserved = 010 : retry + terminate the transaction = 011 : reserved = 100 : acknowledge + terminate the transaction = 101: acknowledge data N = 110 : IACK acknowledge + terminate the transaction = 111 : idle
xpcFLUSH	XPC flushing pending. This signal is activated on IACK flushing detection and remains active until the XPC write buffer are flushed.

Table 4-1 XPC Bus Signals

Signals	*Function*
$xpcBREQ_{xxx}$[01:00]	Individual XPC mastership request
$xpcGNT_{xxx}$	Individual XPC mastership grant

4.1.2 XPC Bus Address Phase

The XPC address phase transports the XPC address and the transactions qualifier. The following bit-mapping is supplied as information:

Table 4-2 XPC Bus Header

Bit	*Name*	*Description*
[01:00]	XADD	Low XPC address (60x bus format)
[31:02]	XADD	XPC address (60x bus format)
[40:32]	EXADD	Extended XPC address (not used on current implementation)
[47:40]	TCNT	Transfer count 64-bit word (data beat)
[48]	DSIZ	Dual size write buffer = 0 : 32 data-beat (256 Bytes) = 1 : 64 data-beat (512 Bytes)
[49]	RFLUSH	Read flush (only used in read transactions) = 0 : read immediate = 1 : read after DIR buffer flushed
[51:50]	APM	Address pre-decode mux control. Hardware support
[54:52]	TSIZ	Transfer size, coded as 60x PowerPC bus
[55]	BURST	Burst transfer Set to 1 for one or more 64-bit data beat Set to 0 for a single data beat of less than 64 bit
[56]	TWRITE	Transaction direction
[57]	SNOOP	Force 60x cache coherency. Only used if TDST= 00
[59:58]	TTYPE	Transfer type = 00 : memory space = 01 : configuration space = 10 : I/O space = 11 : reserved
[61:60]	TSRC	Transaction's source bridge number. Refer to TDST for encoding
[63:62]	TDST	Transaction's destination bridge number = 00 : XPC_PowerPC (system memory) = 01 : XPC_Local_PCI =-10 : XPC_PCI_J3 = 11 : XPC_CompactPCI

4.1.3 XPC Bus Arbitration

The XPC bus is multi-master media shared between four agents. The access to the media is arbitrated through a centralized arbitration with point to point signalling. The pending transaction(s) such as read, singlewrite and dmawrite are handled separately, in order to optimize the media access, sort of QoS. The XPC bus arbitration is pipelined and handled with the following signals:

Table 4-3 XPC Bus Arbitration

Signals	*Function*
$xpcBREQ_{xxx}$[01:00]	XPC access request. Each XPC agent owns these two signals, they are encoded as follows: = 11 : no request = 10 : DMA write = 01 : direct write = 00 : read request
$xpcGNT_{xxx}$	XPC access grant. Each XPC agent owns these two signals.

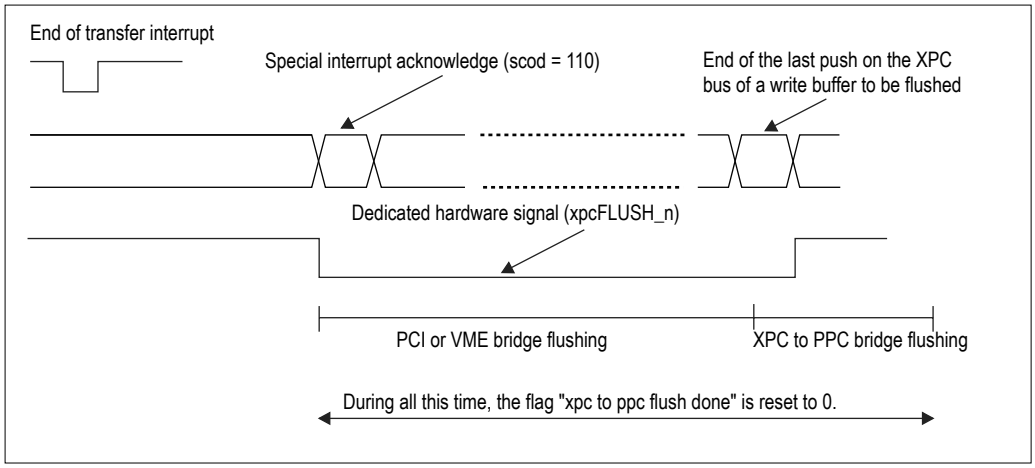
NOTE

The XPC central arbiter is located in the XPC-to-Local_PCI bridge FPGA. The XPC arbitration mode of operation can be selected with the xpcARB_MOD[1:0] located in the LocPCI_CTL register. For more information, please refer to table 8-2.

4.1.4 XPC Buffer Flushing Mechanism

When an XPC master operator (for example a SCSI or an Ethernet controller on a PMC-PCI site) writes data to the SDRAM system memory, there are two levels of write-buffers on the data path (first one in the local PCI-to-XPC bridge and a second one in the CPU-to-XPC bridge). The data can still be in the write-buffer, and not yet in the SDRAM system memory, while the end-of-transfer interrupt occurs. To deal with this delay, a dedicated flushing logic is implemented. This logic uses two extra bits in the interrupt vector to trigger a flushing sequence, which links the specified bridge and a software routine polling the end of flushing through a dedicated status bit.

Figure 4.1 XPC Bus-to-PPC Bus Flushing Mechanism



The interrupt vector (located in the System Interrupt Controller) holds the selected XPC bridge through bits [15:14].

00	No flush to be performed on interrupt
01	Local PCI-to-system memory flush to be performed on interrupt
10	PCI-J3-to-system memory flush to be performed on interrupt
11	CompactPCI-to-system memory flush to be performed on interrupt

The software must poll the bit "xpcppc_Flush_Done" in Local Register_0 (LR0), until this bit is set to 1.

4.1.5 XCSR Bus

The XCSR is a high-speed 8-bit bus used to map all internal XPC_bridge registers. It is implemented with low-level HSTL and runs at 100 MHz. All four XPC bridges incorporate a XCSR Slave port to map their internal registers.

The XCSR Master port is located in the SubI/O FPGA, which can be accessed through the Local_PCI port. Please refer to section 4.1.6 for specific XPC_Bridge's internal registers.

Table 4-4 XPC Bus Arbitration

CPU Address	*XCSR Port*	*Size*
0xF900'9000 - 0xF900'90FF	XPC-to-Local_PCI Bridge	256 Bytes
0xF900'9100 - 0xF900'91FF	XPC-to-PCI_J3 Bridge	256 Bytes
0xF900'9200 - 0xF900'92FF	XPC-to-CompactPCI Bridge	256 Bytes
0xF900'9300 - 0xF900'93FF	PowerPC-to-XPC Bridge	256 Bytes

4.1.6 PowerPC_XPC Internal Registers

The PowerPC-to-XPC bridge incorporates some internal registers. These read / write registers are accessible through the XCSR bus except for XPCPPC_FLUSH, which is directly mapped on the PowerPC bus.

Table 4-5 Internal Registers Mapping

Access from CPU	*Registers*	*Description*
0xF900'9300	XPCPPC_STA	General status information (read only)
0xF900'9320	XPCPPC_CTL	Command and control
0xF900'9340	XPCPPC_DMASIGN	End of DMA chain signature. Read port of FIFO.
0xF900'9360	XPCPPC_XERR	XPC bus address error catch
0xF900'9380	XPCPPC_PERR	PowerPC bus address error catch
0xF900'93C0 - 0xF900'93FC	XPCPPC_DMASTART	Start of DMA chain. Write port of 16 32-bit word deep FIFO
0xFB00'0000	XPCPPC_FLUSH	End flushing status flag

XPCPPC_STA

This read only register provides status information.

Table 4-6 XPCPPC_STA

	PPC_STA CPU = 0xF900'9300		
Bits	***Name***	***Mode***	***Description***
[31:16]	DMA_FSTART_EF	R	Set to 1, when the corresponding FIFO is empty, and reset to 0 when the corresponding FIFO is not empty. [16] for DMA channel_0 [31] for DMA channel_15
[15:13]	DRAM_TYPE	R	This 3-bit field encodes the SDRAM type supported by the FPGA. 000 = 64 Mbit SDRAM (8-bit wide) 001 = 128 Mbit SDRAM (8-bit wide) 010 = 256 Mbit SDRAM (8-bit wide) 011= 256 Mbit SDRAM (4-bit wide)
[12:08]	HW_REV	R	FPGA hardware revision.
[07:04]	Not Used	R	Read as "000"
[03]	DMA_Read_Error	R	Error flag, set if a DMA remote read ends with an error condition
[02]	DMA_Sign_EF	R	DMA signature FIFO empty flag.
[01]	PPC_Error_FL	R	PowerPC bus error flag (SDRAM access out of range). The flag is reset upon access with XPCPPC_PERR register.
[00]	XPC_ERROR_FL		XPC bus error flag (XPC cycle ended with an SCOD = "000"). The flag is reset upon access with XPCPPC_XERR register.

XPCPPC_CTL

This read write register provides command and control.

Table 4-7 XPCPPC_CTL

	PPC_CTLCPU = 0xF900'9320		
Bits	***Name***	***Mode***	***Description***
[31:16]		R	
[12]	SDRAM_REF	R/W	SDRAM controller configuration: 0 = 4K refresh 1 = 8K refresh
[11]	SDRAM_PARITY	R/W	SDRAM controller configuration: 0 = parity not provided 1= parity provided
[10]	SDRAM_BLOCK	R/W	SDRAM controller configuration: 0 = piggy with 1 block (9 SDRAM devices) 1= piggy with 2 blocks (18 SDRAM devices)
[09:08]	SDRAM_TYPE	R/W	SDRAM controller configuration: 00 = 64 Mbit (8-bit wide devices) 01 = 128 Mbit (8-bit wide devices) 10 = 256 Mbit (8-bit wide devices) 11 = 256 Mbit (4-bit wide devices)
[07]	CPCI_2G	R/W	This bit, while set, allows to provide 2 GBytes of physical addressing over CompactPCI. In this case the XPC_PCI_J3 is disabled
[06]	SNOOP_DIS	R/W	Disable cache snooping. Must be left at 0 in normal operation
[05]	DMA_IRQENA_RERR	R/W	Enable DMA interrupt on DMA remote read error detection
[04]	DMA_IRQENA_EF	R/W	Enable DMA interrupt on normal termination (DMA signature FIFO not empty)
[03]	Not Used	R/W	CES reserved
[02]	PARITY_Test	R/W	Invert the parity generation on the PowerPC. Used to test the parity function. Must be left at 0 in normal operation
[01]	PARITY_Ena	R/W	Enable the parity protection mechanism. Must be set at boot time only
[00]	DMA_Ena	R/W	Enable the embedded DMA controller core

NOTE

The five SDRAM configuration bits (PPC_CTL [12:08]) are read from the SubI/O component (SUBIO_REG2 register, F9008008, ([04:00]) and written at boot time in the PPC-to-XPC Bridge XPCPPC_CTL register, without any alteration.

XPCPPC_DMASIGN

Please refer to chapter 10 for a complete description of this register.

XPCPPC_XERR

This register catches the XPC address on which an XPC bus error occurred (SCOD="000"). A boolean is then set to "1" in the XPCPPC_STA register "XPC_ERROR_FL" to indicate that the register contains a faulty address.

The boolean "XPC_ERROR_FL" is reset when the XPCPPC_XERR register is accessed. This mechanism catches all errors which occur on the XPC bus.

Table 4-8 XPCPPC_XERR

	PPC_XERRCPU = 0xF900'9360		
Bits	***Name***	***Mode***	***Description***
[31:00]	XERR_ADD	R	XPC address A[31:00] caught

XPCPPC_PERR

This register catches the PowerPC address, on which a TEA (PowerPC Bus Transfer Error Acknowledge) occurred. A boolean is then set to "1" in the XPCPPC_STA register "PPC_ERROR_FL" to indicate that the register contains a faulty address.

The boolean "PPC_ERROR_FL" is reset when the XPCPPC_PERR register is accessed. This mechanism catches all errors occurring on the PowerPC bus.

Table 4-9 XPCPPC_PERR

	***PPC_PERR**CPU = 0xF900'9380*		
Bits	***Name***	***Mode***	***Description***
[31:00]	PERR_ADD	R	PowerPC address A[31:00] caught

XPCPPC_DMAFIFO

Please refer to chapter 10 for a complete description of this register.

XPCPPC_FLUSH

This register is not mapped over the XCSR bus, but directly on the PowerPC bus. This provides a fast, non-disturbing mechanism for continuous polling by the CPU.

This bit should be polled by software after an interrupt acknowledge. By doing so, the software is able to know, when data is truly available in the system memory, after a posted write is performed by an XPC master.

NOTE

An Ethernet controller will receive some data from the network, write them to the system memory over the XPC bus, and signal, by an interrupt, that new data is available to the CPU. Although, at interrupt time, data may still be "on the way" to the system memory. It may be in one of the write buffers of the bridges, and not yet in the system memory. By polling this bit, after an interrupt, software will know when data is truly readable from system memory.

Please refer to section 4.1.4 for a more detailed explanation of the XPC flushing mechanism.

Table 4-10 XPCPPC_FLUSH

	***PPC_FLUSH**CPU = 0xFB00'0000*		
Bits	***Name***	***Mode***	***Description***
[31:01]	Not Used	R	Reserved
[00]	WB_FLUSH_DONE	R	This status is set to 1 when the flushing hardware sequence is terminated

4.1.7 Direct Access to Internal Registers

The preferred pathway to access internal registers is the XCSR bus one, described in previous section. But, for time critical applications, a quicker "direct" pathway is provided to access PowerPC-XPC internal registers. It is software responsibility to not mix XCSR access and Direct access to the PowerPC XPC internal registers. This may lead to unpredictable results.

The "direct" mapping is described in the following table:

Table 4-11 Internal Registers Direct Mapping

Access from CPU	***Registers***	***Description***
0xFB00'3300	XPCPPC_STA	General status informations (read only)
0xFB00'3320	XPCPPC_CTL	Command and control
0xFB00'3340	XPCPPC_DMASIGN	End of DMA chain signature. Read port of FIFO.
0xFB00'3360	XPCPPC_XERR	XPC bus address error catch
0xFB00'3380	XPCPPC_PERR	PowerPC bus address error catch
0xFB00'33C0 - 0xFB00'33FC	XPCPPC_DMASTART	Start of DMA chain. Write port of 16 32-bit word deep FIFO.
0xFB00'0000	XPCPPC_FLUSH	End flushing status flag

Chapter 5
Board Resources

5.1 Onboard Local Resources

The Local_PCI (64-bit at 33 MHz) is the internal PCI bus which connects the onboard PCI resources of the board, as well as the two onboard PMCs. Please refer to chapter 8 for a complete description of XPC-to-Local_PCI Bridge and directly connected resources such as Ethernet Controller and PMC slots.

The other local resources are mapped onto the upper 128 MBytes of the Local_PCI address space, but are physically interfaced with a dedicated FPGA, called SubI/O.

Figure 5.1 XPC-Local_PCI and SubI/O Resource Block Diagram

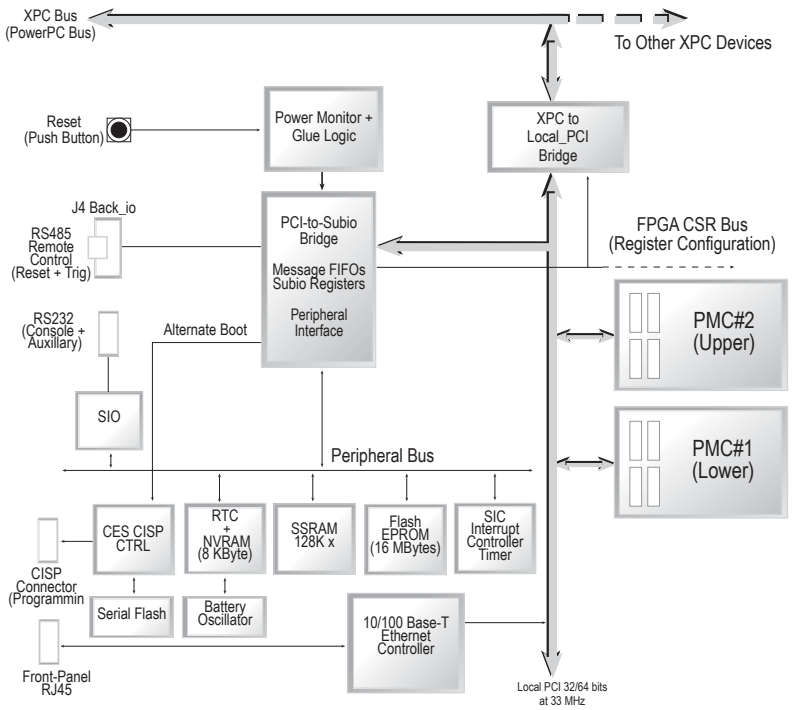


Table 5-1 SubI/O Resources Mapping

CPU Address	*Local_PCI Address*	*Resources*
0xFC00'0000-0xFFFF'FFFF	$0xFC00'0000_{MEM}$-$0xFFFF'FFFF_{MEM}$	Flash EPROM
0xFA10'0000-0xFA1F'FFFF	$0xFA10'0000_{MEM}$-$0xFA1F'FFFF_{MEM}$	RTC + NVRAM

Table 5-1 SubI/O Resources Mapping

CPU Address	*Local_PCI Address*	*Resources*
0xFA00'0000-0xFA0F'FFFF	$0xFA00'0000_{MEM}$-$0xFA0F'FFFF_{MEM}$	SSRAM 128K x 32
0xF900'E000-0xF900'FFFF	$0xF900'E000_{MEM}$-$0xF900'FFFF_{MEM}$	FIFO Port Registers
0xF900'C000-0xF900'DFFF	$0xF900'C000_{MEM}$-$0xF900'DFFF_{MEM}$	SIO Register
0xF900'B000-0xF900'BFFF	$0xF900'B000_{MEM}$-$0xF900'BFFF_{MEM}$	FIFO Control Registers
0xF900'9000-0xF900'9FFF	$0xF900'9000_{MEM}$-$0xF900'9FFF_{MEM}$	FPGA CSR Bus Registers
0xF900'8000-0xF900'8FFF	$0xF900'8000_{MEM}$-$0xF900'8FFF_{MEM}$	SubI/O Registers
0xF900'4000-0xF900'5FFF	$0xF900'4000_{MEM}$-$0xF900'5FFF_{MEM}$	Timer
0xF900'0000-0xF900'1FFF	$0xF900'0000_{MEM}$-$0xF900'1FFF_{MEM}$	SIC Interrupt Controller
0xF800'0000-0xF8FF'FFFF	$0xF800'0000_{MEM}$-$0xF8FF'FFFF_{MEM}$	Special D16 Flash Programming

WARNING

The un-documented address field should not be accessed, due to the undecoded field, which could hang the processor for ever.

5.1.1 Flash EPROM

The RIOC 4068 is configured with 16 MBytes of Flash EPROM. The Flash EPROM is used for storing the start-up firmware (PPCMon), an OS and a user application. Compression mechanisms increase the storage capacity of the Flash EPROM by a factor of up to three.

Table 5-2 Flash EPROM Address Space

CPU	*Sector Name*	*Description*
0xFF00'0000 - 0xFF0F'FFFF	Boot	CPU boot code
0xFF10'0000 - 0xFFFF'FFFF	User Free	Free for user

The Flash EPROM can be online programmed through RS232 or ethernet downloading. Flash EPROM programming functions are available from PPCMon and from the OS. Please refer to the corresponding documentation for more information.

5.1.2 RTC / NVRAM

Both non-volatile RAM and a Real-Time Clock RTC are contained in the NVRAM (SGS-Thomson, M48T59). The NVRAM has its own lithium battery to operate the clock and to maintain the contents of the NVRAM during power OFF situations. The battery device provides backup for approximately seven years. The Real-Time Clock circuitry provides accuracy of plus or minus one second per day.

WARNING

There is a danger of explosion if the battery is incorrectly replaced. Replace only with the same battery (**SGS Thomson M4T28 BR12-SH 1**) or an equivalent battery recommended by the manufacturer. Dispose of used batteries according to the manufacturer's instructions.

The RTC / NVRAM device must be accessed in 32-bit words, although valid data is only available on the lower byte (bits [07:00]).

RTC REGISTERS MAPPING

Table 5-3 RTC Registers Mapping

CPU Address	*Register Name*
0xFA10'7FC0	RTC Flags
0xFA10'7FC4	RTC Not Used
0xFA10'7FC8	RTC Alarm Seconds
0xFA10'7FCC	RTC Alarm Minutes

Table 5-3 RTC Registers Mapping

CPU Address	*Register Name*
0xFA10'7FD0	RTC Alarm Hours
0xFA10'7FD4	RTC Alarm Date
0xFA10'7FD8	RTC Interrupts
0xFA10'7FDC	RTC Watchdog
0xFA10'7FE0	RTC Control
0xFA10'7FE4	RTC Seconds
0xFA10'7FE8	RTC Minutes
0xFA10'7FEC	RTC Hours
0xFA10'7FF0	RTC Day
0xFA10'7FF4	RTC Date
0xFA10'7FF8	RTC Month
0xFA10'7FFC	RTC Year

RTC CALIBRATION

The PPCMon firmware provides facilities to run calibrations for the RTC.

RTC RESET

The Real-Time Clock controller provides power monitoring of the + 5 Volts power supply. Under 4.75 Volts (typical) an internal logic switches to the battery backup mode, and protects the NVRAM against write-access. At the same time, the RTC chip asserts its reset signal request to the internal reset logic.

RTC WATCHDOG

The RTC provides an internal 8-bit Watchdog Timer, which can generate either a board reset, or a CPU interrupt through the interrupt controller. The Watchdog Timer can be programmed to generate time-out. The PPCMon firmware and the CES BSPs provide facilities to handle the RIOC 4068 Watchdog function.

NVRAM

Table 5-4 NVRAM Mapping

CPU	*Register Name*
0xFA10'0000 - 0xFA10'3FFC	User Free
0xFA10'4000 - 0xFA10'7FFC	CES-Reserved

5.1.3 Timers

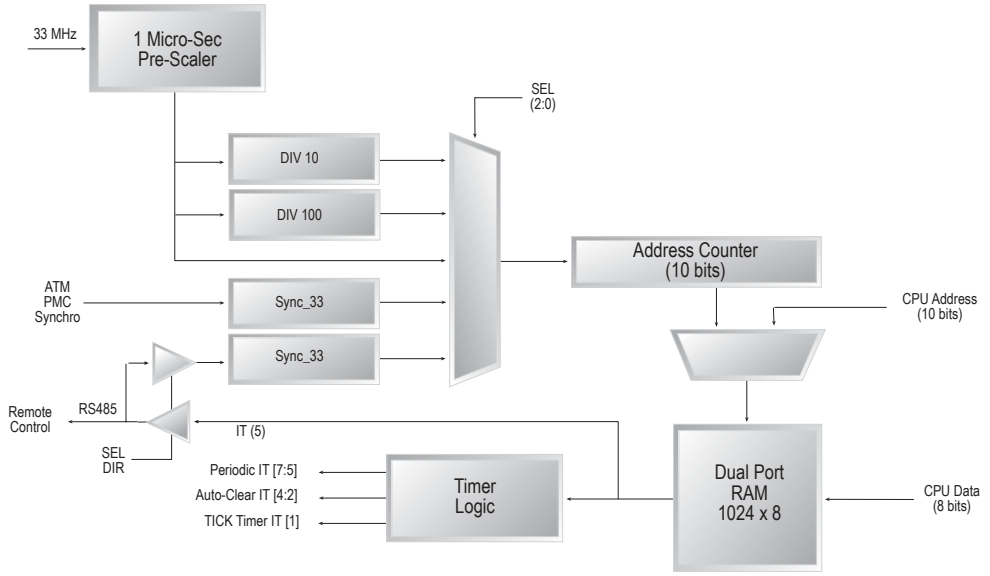
The RIOC 4068 offers onboard timing facilities. Built within an FPGA internal dual-port RAM (1024 x 8-bit), the timer can be clocked from a pre-scaler by selecting one of the following clock sources:

- Internal 1 µs period clock
- Internal 10 µs period clock
- Internal 100 µs period clock
- External RS485 signal (front panel connector) connected to J4 back_io
- CES ATM PMC synchro signal (specific 8 KHz)

FEATURES

- Timer roll-over for maximum 1 ms, 10 ms, 100 ms with the internal clock
- Counter clock granularity: 1 µs, 10 µs, 100 µs (internal clock), ATM PMC synchro signal, external signal
- Filtered and synchronized external count increments for RS485 trigger input or ATM synchro
- Six independent (3 x periodic / 3 x auto-clear) trimmed interrupts for user-timing event purposes
- Interrupt generation when the counter reaches zero with an automatic reload for the time-out process

Figure 5.2 Timer Block Diagram



TIMER REGISTERS MAPPING

The timer registers are mapped at the following addresses:

Table 5-5 Timer Registers Mapping

CPU	*Data*	*Reg. Name*	*Description*
0xF900'4000 - 0xF900'4FFC	8 bits	TIMER_RAM	R/W dual-port RAM 1024 x 8
0xF900'5000	8 bits	TIMER_REG	R/W timer command register
0xF900'5010	10 bits	TIMER_ACN	R timer counter latched address

The 8 bits of data are used to generate the following signals:

Table 5-6 TIMER RAM Data Description

Bits	*Name*	*Mode*	*Description*
[07:05]	ARIR	R/W	3 bits are user-free for auto-reloaded interrupt request (periodic IT)
[04:02]	ACIR	R/W	3 bits are user-free for auto-cleared interrupt request (single shot IT)
[01]	ZD_TICK	R/W	Reserved for the zero detect / TICK Timer (periodic IT)
[00]	RLD_ROD	R/W	Reserved for the re-load and the roll-over detection

Table 5-7 Timer Counter Latched Address Register

Bits	*Name*	*Mode*	*Description*
[07]	PRG_CLR	R/W	Clears the timer address counter when set to 1.
[06:05]	Not Used	R	Read as zero
[04]	RS485_DIR	R/W	Selects RS485 I/O direction (0 at reset): 0 = input 1 = output

Table 5-7 Timer Counter Latched Address Register

Bits	*Name*	*Mode*	*Description*
[03:01]	DIV_SEL	R/W	DIV_SEL selects the timer clock frequency increment: 000 = 10 µs count increments 001 = 100 µs count increments 010 = 1 µs count increments 011 = external RS485 count increments 100 = PMC ATM synchro (8 KHz) increments
[00]	TIME_RUN	R/W	Starts the timer counter when 1, or stops when 0.

NOTE The timer must be stopped (bit [00]) before being cleared (bit [07]).

RS485 TIMER TRIGGER

The J4 back panel connector (user defined) provides one bi-directional RS485 channel connected both to the SIC (System Interrupt Controller) and to the timer unit. The RS485 channel direction can be selected through bit [04] of the timer counter register.

TRIGGER OUTPUT

For output, the RS485 channel is connected to the TIMER_RAM bit [05]. The pulse length and the minimum / maximum frequency depends on the timer increment source.

TRIGGER INPUT

For input, the RS485 is used to increment the timer address counter, when selected by the timer command register.

WARNING For the Trigger input, the minimum pulse length is 150 ns.
The CompactPCI Back-io module is optional and it could be designed for customized applications

PMC ATM SYNCHRO

The 8 KHz signal is a 50 ns synchro-pulse every 124,950 ns driven onto the CES ATM family of PMCs from the PMC user I/O connector (PN14-pin36).

The RIOC 4068 connects this signal from JN24-pin36 (upper PMC#2) and from the JN14-pin36 (upper PMC#1).

ADDRESS ZERO INTERRUPT

Since the first location (address = 0x00) of the TIMER_RAM does not contain any interrupts (data = 0x00), the address zero detect interrupt should be set by the software at the last RAM location (where the RAM bit [00] is set) by using a periodic interrupt such as the TICK Timer (RAM bit [01]).

ROLL-OVER INTERRUPT

The roll-over interrupt (TIMER_RAM bit [00]) must be set by the software at the last RAM initialized location.

WARNING The auto-clear interrupts (RAM bits [04:02]) must not be set at the roll-over location, otherwise the timer logic will loop forever at zero.

PERIODIC SIC INTERRUPTS TIME

The three periodic SIC interrupts are generated by the timer logic every time the TIMER_RAM bits [07:05] are set to 1. The timer roll-over will reassert these periodic interrupts.

AUTO-CLEAR INTERRUPTS TIME

The three auto-clear interrupts are generated by the timer logic every time the TIMER_RAM bits [04:02] are set to 1. These bit locations are cleared when the timer rolls over.

The CES BSPs contain high-level functions to control the board timers.

5.1.4 Ethernet

Please refer to section 8.2 for more details.

5.1.5 SIO Serial Ports

Two serial interfaces are available to the user on the front panel µDB9 connector. Port A is labelled “Console” (“CONS.”) and Port B is labelled “Auxiliary” (“AUX.”).

The serial ports A and B provide RS232 with a full asynchronous modem with up to 1 Mbaud support.

The device used to provide these two serial lines is the EXAR ST16C2550 chip.

PORT A AND PORT B MAPPING

Table 5-8 Mapping for Port A and Port B

CPU	*SIO Port*
0xF900’C400	A (console)
0xF900’C000	B (auxiliary)

REGISTER SELECTION

The addresses given below are offset from the port base address.

Table 5-9 Internal Registers Decoding

Offset	*Read*	*Write*	*Conditions*
0x00	Receive Holding Register	Transmit Holding Register	LCR [07] = 0
0x04	Interrupt Enable Register	Interrupt Enable Register	LCR [07] = 0
0x08	Interrupt Status Register	FIFO Control Register	None
0x0C	Line Control Register	Line Control Register	None
0x10	Modem Control Register	Modem Control Register	None
0x14	Line Status Register	N/A	None
0x18	Modem Status Register	N/A	None
0x1C	Scratchpad Register	Scratchpad Register	None
0x00	LSB of Divisor Latch	LSB of Divisor Latch	LCR [07] = 1
0x04	MSB of Divisor Latch	MSB of Divisor Latch	LCR [07] = 1

Please refer to the EXAR ST16C2550 data sheet for device programming.

RS232C PORTS

The front panel RS232 is available through the µDB9 connectors (ITT Cannon MDSM double- decker MDSM-18 P E-Z7).

The pin assignment is identical to the standard µDB9.

Table 5-10 Pin Assignment

Connector Pin #	*Signal Name*	*Description*
pin #1	DCD	Data carrier detect
pin #2	SIN	Serial input
pin #3	SOUT	Serial output
pin #4	DTR	Data terminal ready
pin #5	GND	Signal ground
pin #6	DSR	Data set ready

Table 5-10 Pin Assignment

Connector Pin #	*Signal Name*	*Description*
pin #7	RTS	Request to send
pin #8	CTS	Clear to send
pin #9	RI	Ring indicator

NOTE 15 KV ESD protection is provided on both RS232 channels.

5.1.6 SSRAM

The local SSRAM (512 KBytes) is divided in two areas:
- Data storage for the general purpose scratchpad (free for the user)
- Data storage for the eight message passing FIFOs (reserved)

Table 5-11 SSRAM

CPU	*Register Name*
0xFA00'0000 - 0xFA05'FFFF	Scratchpad SRAM
0xFA06'0000 - 0xFA07'FFFF	FIFO Storage

SCRATCHPAD STORAGE

This memory area can be accessed by the user-application as a general purpose scratchpad SRAM.
It supports read and write accesses in burst mode to increase the access time performance.

FIFO STORAGE

This area should not be directly accessed by the user application.

5.1.7 Message Passing FIFOs

Eight message passing FIFOs are implemented in the RIOC 4068. Each FIFO is 4096 words deep and 32 bits wide.

The eight FIFOs work individually. Each one drives two flags: full and empty. These two flags are monitored by the SIC (System Interrupt Controller) to interrupt the CPU under the following conditions:

- FIFO NOT EMPTY Indicates that something is present in the FIFO
- FIFO EMPTY Indicates that the FIFO is empty
- FIFO FULL Indicates that the last message has been caught. The next write will generate an error.

The FIFOs are generally used in multiprocessor environments to provide:

- Message passing
- Virtual interrupts
- Resource sharing
- Task calling sequence.

The eight FIFOs can be used individually, allowing to define priority between the queues (for message passing) or to define one FIFO for message passing and another one for data buffer reservation.

Reading or writing to a FIFO is done through the FIFO port register. The data is then automatically read or written to the FIFO message storage memory. The FIFO word counter, FIFO read pointer and FIFO write pointer are updated by the hardware logic.

The FIFO message storage memory, FIFO Word Counter, FIFO Read Pointer and FIFO Write Pointer should be accessed directly only for debugging purposes.

Table 5-12 FIFOs Ports Mapping

CPU	*Register Name*
0xF900'E000	FIFO#0
0xF900'E400	FIFO#1
0xF900'E800	FIFO#2
0xF900'EC00	FIFO#3
0xF900'F000	FIFO#4
0xF900'F400	FIFO#5
0xF900'F800	FIFO#6
0xF900'FC00	FIFO#7

Table 5-13 FIFO Message Storage Memory Mapping

CPU	*Register Name*
0xFA06'0000 - 0xFA06'3FFF	FIFO#0 Storage
0xFA06'4000 - 0xFA06'7FFF	FIFO#1 Storage
0xFA06'8000 - 0xFA06'BFFF	FIFO#2 Storage
0xFA06'C000 - 0xFA06'FFFF	FIFO#3 Storage
0xFA07'0000 - 0xFA07'3FFF	FIFO#4 Storage
0xFA07'4000 - 0xFA07'7FFF	FIFO#5 Storage
0xFA07'8000 - 0xFA07'BFFF	FIFO#6 Storage
0xFA07'C000 - 0xFA07'FFFF	FIFO#7 Storage

Table 5-14 FIFO Control Registers

CPU	*Register Name*
0xF900'B000	FIFO#0 Word Counter
0xF900'B004	FIFO#0 Write Pointer
0xF900'B008	FIFO#0 Read Pointer
0xF900'B010	FIFO#1 Write Pointer
0xF900'B014	FIFO#1 Word Counter
0xF900'B018	FIFO#1 Read Pointer
0xF900'B020	FIFO#2 Write Pointer
0xF900'B024	FIFO#2 Word Counter
0xF900'B028	FIFO#2 Read Pointer
0xF900'B030	FIFO#3 Write Pointer
0xF900'B034	FIFO#3 Word Counter
0xF900'B038	FIFO#3 Read Pointer
0xF900'B040	FIFO#4 Write Pointer
0xF900'B044	FIFO#4 Word Counter
0xF900'B048	FIFO#4 Read Pointer
0xF900'B050	FIFO#5 Write Pointer

Table 5-14 FIFO Control Registers

CPU	*Register Name*
0xF900'B054	FIFO#5 Word Counter
0xF900'B058	FIFO#5 Read Pointer
0xF900'B060	FIFO#6 Write Pointer
0xF900'B064	FIFO#6 Word Counter
0xF900'B068	FIFO#6 Read Pointer
0xF900'B070	FIFO#7 Write Pointer
0xF900'B074	FIFO#7 Word Counter
0xF900'B078	FIFO#7 Read Pointer

5.1.8 Thermometer

The RIOC 4068 board implements three temperature sensors to monitor the temperature on the following devices:
- PowerPC processor
- PMC#1
- PMC#2

CPU THERMOMETERS

The PowerPC 7455 features an internal thermometer (TAU), which monitors the temperature of the silicon but this feature is no more supported by IBM and MOTOROLA (refer to the IBM / Motorola documentation).

For this reason, the RIOC 4068 implements an additional thermometer located under the CPU Heatsink

PMC'S THERMOMETERS

These thermometers are located in the middle of the nose of the PMC#1 and PMC#2 in order to control the temperature between the motherboard and these hardware plug-ins.

The access to the serial thermometers is controlled by the register SUBIO_REG#6 through a serial bus (2-wires) at the following Thermometers strapped addresses:
- Address_0 : PMC#1 none thermometer
- Address_1 : PMC#2 none thermometer
- Address_2 : CPU heatsink thermometer.

THERMOMETER INTERRUPT

Only one line of interrupt request is connected to the System Interrupt Controller. This IRQ is common to the two thermometer devices and it is the responsibility of the software to determine which thermometer has asserted the interrupt line.

Table 5-15 Thermo IRQ Register Mapping

CPU	*Register Name*
0xF900'0500	THERMO IRQ Register

Table 5-16 THERMO IRQ Register

Bits	*Name*	*Mode*	*Description*
[07:03]	Not Used	R-Only	Read as "zero"
[02]	CPU_TEMP_CTL	R-Only	CPU Heatsink temperature control
[01]	PMC#2_TEMP_CTL	R-Only	Lower PMC#2 temperature control
[00]	PMC#1_TEMP_CTL	R-Only	Lower PMC#1 temperature control

5.1.9 Watchdog Timer

The RIOC 4068 uses the RTC device to provide the Watchdog Timer function. The Watchdog Timer can either generate an interrupt to the CPU or generate a global reset of the board.

The Watchdog Timer can be used to detect an out-of-control microprocessor. The user programs the Watchdog Timer by setting the desired amount of time-out into the 8-bit Watchdog register.

Table 5-17 Watchdog Timer Mapping

CPU	*Register Name*
0xFA10'7FDC	Watchdog register

Table 5-18 Watchdog Timer Register

Bits	*Name*	*Mode*	*Description*
[07]	WDS	R/W	Watchdog steering bit. When set to 1, the Watchdog will generate a reset, otherwise it will generate an IRQ.
[06:02]	BMBx	R/W	Binary multiplier. The amount of the time-out is determined by the multiplication of this multiplier with the resolution ([01:00]).
[01:00]	RBx	R/W	Resolution: 00 = 1/16 second 01 = 1/4 second 10 = 1 second 11 = 4 seconds

EXAMPLE Writing 00001110 in the Watchdog register = 3 x 1 or 3 seconds.

If the processor does not reset the timer within the specified period, the Watchdog Timer generates a Watchdog interrupt or a cold reset.

5.1.10 SubI/O Local Registers

The local registers are located into the SubI/O FPGA and manage the basic SubI/O functions.

Table 5-19 Local Registers

CPU	*Reg. Name*	*Register Function*
0xF900'8000	SUBIO_REG#0	Status register RS485 control and programmed-reset
0xF900'8004	SUBIO_REG#1	Flash EPROM device programming
0xF900'8008	SUBIO_REG#2	SDRAM status register (read-only)
0xF900'800C	SUBIO_REG#3	Glue logic status register (read-only)
0xF900'8010	SUBIO_REG#4	CES In-Situ Programming register (CISP CPU port)
0xF900'8014	SUBIO_REG#5	CES-reserved for maintenance bus interface
0xF900'8018	SUBIO_REG#6	SubI/O LED + temperature controllers
0xF900'801C	SUBIO_REG#7	SubI/O FPGA revision register (read only)

SUBIO_REG#0 (RESET & RS485 CONTROL)

Table 5-20 SUBIO_REG#0 Register

	SUBIO_REG#0$_{CPU}$ = 0xF900'8000		
Bits	***Name***	***Mode***	***Description***
[31:08]	Not Used		
[07]	PROG_RESET	W-Only	Setting this bit to 1 resets the board (warm reset)
[06]	CH#1_SDI	R	RS485 channel#1 data in
[05]	CH#1_SDO	R/W	RS485 channel#1 data out
[04]	CH#1_DIR	R/W	Channel#1 data direction: 1 = Tx mode 0 = Rx mode
[03]	CH#2_SDI	R-Only	RS485 channel#2 data in
[02]	CH#2_SDO	R/W	RS485 channel#2 data out
[01]	CH#2_DIR	R/W	Channel#2 data direction: 1 = Tx mode 0 = Rx mode
[00]	BOOT_SWITCH	R-Only	Reflects the status of the boot DIP switch: SW401-3 1 = switch OFF: standard boot onto Flash EPROM 0 = switch ON: alternate boot onto serial flash

The two RS485 channels are routed to the front panel µDB9 connector P403.

SUBIO_LOC_REG#1 (FLASH EPROM PROGRAMMING)

Table 5-21 SUBIO_REG#1 Register

	SUBIO_REG#1$_{CPU}$ = 0xF900'8004		
Bits	***Name***	***Mode***	***Description***
[31:04]	Not Used		Bits [31:08] are Undefined "X" Bits [07:04] are Always "ZERO"
[03]	Prog_Enable	R/W	Enable Flash EPROM programming 1 = Enable 0 = Disable
[02]	RESERVED	R-Only	Bit [02] is Always "ZERO"
[01:00]	Select_Flash	R/W	Select Flash-Eprom bank to program 11 = Reserved 01 = Program bank #1 10 = Program bank #2 00 = Disable programming

The Flash EPROM programming use a special D16 interface. The PPCMon firmware manages that directly. For more information, please refer to the PPCMon User's Manual (Ref. DOC 3317C/UI).

SUBIO_REG#2 (SDRAM INFORMATION)

Table 5-22 SUBIO_REG#2 Register

	SUBIO_REG#2$_{CPU}$ = 0xF900'8008		
Bits	***Name***	***Mode***	***Description***
[31:05]	Not Used		Bits [31:08] are undefined "X" Bits [07:05] are always "ZERO"
[04:00]	CES Reserved for SDRAM Piggyback	R-Only	This five bit field provides the CES specific codes for the different memory piggyback.

The SDRAM types are defined by using hardware-straps located onto the piggyback memory, which allows automatic recognition by the PPCMon firmware.

SUBIO_REG#3 (HARDWARE SETTING INFORMATION)

Table 5-23 SUBIO_REG#3 Register

	SUBIO_REG#3$_{CPU}$ = 0xF900'800C		
Bits	***Name***	***Mode***	***Description***
[31:08]	Not Used		Bits [31:08] are undefined "X"
[07:04]	Cpu_Vdd_Core	R-Only	Status for the setting of the CPU Vdd_Core: 1111 = 1.30 Volts ..... 0001 = 1.95 Volts 0001 = 2.00 Volts 0000 = 2.05 Volts
[03]	PCI_J3_CLK	R-Only	Status for setting of the J3 Clock: 0 = Select 33 MHz 1 = Select 25 MHz
[02:00]	Clock_PLL_Config	R-Only	Status for setting of the CPU and the XPC bus: 111 = 100MHz 011 = 66MHz 101 = 90MHz 001 = 60MHz 110 = 83 MHz 010 = 50MHz 100 = 70MHz 000 = 40MHz

These hardware-settings are defined by programming the CPLD_01.

SUBIO_REG#4 (CISP PROGRAMMING)

Table 5-24 SUBIO_REG#4 Register

	SUBIO_REG#4$_{CPU}$ = 0xF900'8010		
Bits	***Name***	***Mode***	***Description***
[31:08]	Not Used		
[07]	Cisp_Busy	R-Only	Reflects the status of the CISP controller interface 1 = Busy 0 = Ready
[06]	Cisp_CS	R/W	Chip select control bit for the CISP interface 1 = Assert CS# signal 0 = Deassert CS# Signal
[05]	Cisp_CLK	R/W	Serial clock control bit for the CISP interface 1 = Assert CLK signal 0 = Deassert CLK Signal
[04]	Cisp_SDI	R-Only	Serial data in from the CISP interface
[03]	Cisp_SDO	W-Only	Serial data out to the CISP interface
[02:00]	Not Used	R-0nly	Always read as "ZERO"

The Loc_REG#4 allows the CPU to access the CES In Situ Programming (CISP) Interface in order to re-program the Serial-flash EPROM which stores the FPGA logic, the primary boot and the board signature (CES reserved). The PPCMon firmware provides routines to access this resource.

SUBIO_REG#5 (RESERVED)

Table 5-25 SUBIO_REG#5Register

	$SUBIO\_REG\#5_{CPU}$ = 0xF900'8014		
Bits	***Name***	***Mode***	***Description***
[31:00]	Not Used		
[07:00]	Reserved	R	Read as "Zero"

NOTE

This register is not used, but it is reserved by CES for future use.

SUBIO_REG#6 (TEMPERATURE CONTROLLER)

Table 5-26 LOC REG#6 Register

	$SUBIO\_REG\#6_{CPU}$ = 0xF900'8018		
Bits	***Name***	***Mode***	***Description***
[31:08]	Not Used		
[07]	SUBIO_LED	R/W	Programmable front panel LED T0
[06:04]	Not Used	R	Read as "ZERO"
[03]	TEMP_SDI	R	Serial data in from the temperature controllers (wired OR)
[02]	TEMP_RW	R/W	Data direction on the temperature control serial bus (2-wire bus I/F): 1 = write 0 = read
[01]	TEMP_SDO	R/W	Serial data out from the temperature controllers (wired OR)
[00]	TEMP_SCL	R/W	Serial clock

NOTE

- A temperature controller uses a two-wire bus. Please refer to the TELCOM TCN7533 documentation for a programming guide.
- Each temperature controller has it own address:
  - Address 0 = Lower PMC#1 temperature controller
  - Address 1 = Upper PMC#2 temperature controller
  - Address 2 = CPU Heatsink temperature controller

5.2 PMCs Slot

Please refer to section 8.3.

5.3 Onboard DC / DC Power Supplies

The RIOC 4068 family boards provides some onboard DC / DC's and Drop-Down for the following supplies:

- 2.5 Volts power for the LVTTL fast I/O and the L3 core
- 1.8 Volts power for the FPGA core
- 1.5 Volts power for the HSTL III I/O
- 1.5 Volts power for the L3 very fast I/O
- 1.3 Volts power for the CPU_Core (PowerPC Spec N)

5.3.1 CPU Core Power

The CPU core power supply is implemented by using a programmable DC / DC which steps down the + 5 Volts for a voltage range between 1.00 to 2.00 Volts and a current max. of 15 Amps. The CPU core voltage is selected by 2 resistors divider that manages the different PowerPC processor voltage requirements.

L3 AND CPU VDDQ POWER

The interface between the CPU and the L3-DDRAM use a LDO device that drop down the 2.5 Volts to supply the Vddq power 1.5 Volts /1.5 A for the CPU and the L3 cache.

L3 CORE POWER

The onboard 2.5 Volts DC/DC supplies the power for the L3 cache DDR-SRAM Vdd core.

5.3.2 2.5 Volts Power

The 2.5 Volts DC / DC steps down the + 5 Volts and supplies up to 7.5 Amps for the LVTTL fast I/O (between the CPU and the FPGA CPU-Bridge) and the L3 Cache Vdd_Core.

NOTE

The LVTTL fast I/O between the PowerPC processor, and the FPGA CPU bridge uses the 2.5 Volts signaling to increase buses performance.
The 2 onboard Drop-Down use this 2.5V power to supply the 1.5 Volts for HSTL and L3_i/o

5.3.3 1.8 Volts Power

The 1.8 Volts DC / DC step down the + 5 Volts Power and supply up to 7.5 Amps for the FPGA Vdd_Core (Virtex_E).

5.3.4 1.5 Volts Power

The 1.5 Volts LDO drop down the + 2.5 Volts Power and supply up to 1.5 Amps for the HSTL III signaling. The HSTL signaling is used for the 100 MHz XPC bus and CSR bus.

5.4 Live Insertion Controller

The Live Insertion controller (L/I) provides over current protection, power good detection and power on / down rampling control. Please refer to the LTC1643L Linear-Technology Data Sheet for more information.

5.4.1 Board Power ON / OFF

The Front panel bottom handles control the switch ON / switch OFF of the board through the ON input pin of the L/I controller.

5.4.2 Over Current Protection

The current for the CompactPCI backplane power supplies is controlled and limited by the L/I controller. When an over current condition is detected the L/I controller switches OFF the power on the board and asserts the over current signal, which lights on an internal Red_LED located on the bottom PCB side close to the P2 connector.

Table 5-27 Over Current Control

CompactPCI Power Supply	*Maximum Value*
+ 5 Volts	15 Amps
3.3 Volts	15 Amps
+ 12 Volts	850 mili-Amps
- 12 Volts	450 mili-Amps

WARNING When the over current occurs, the condition is latched until the board is powered OFF then powered ON again and the current is again into the specified range.

5.4.3 Power Good Detection

The power good signal is asserted by the L/I controller when the power supplies, from the CompactPCI backplane, rise above the following values.

Table 5-28 Power Good Control

Power Supply	*Minimum Value*
+ 5 Volts	4.75 Volts
3.3 Volts	3.15 Volts
+ 12 Volts	11.4 Volts
- 12 Volts	- 10.8 Volts

NOTE The onboard DC / DC power supplies are not monitored by the L/I Controller power good signal, but by an onboard power-monitor

5.4.4 Power Rampling Control

When the CompactPCI + 12 Volts rises over + 9 Volts, the L/I controller switch on the FET transitors for the + 5 Volts and the 3.3 Volts. Then the Power will ramp up around 10 milli-second.

WARNING The CompactPCI + 5V and 3.3V power should reach the CompactPCI minimum value when the CompactPCI + 12 V power rises over + 9V in order to guaranty the minimum L/I power-up ramp.

5.5 RESET System

The central reset logic is implemented by using a Lattice 2128VE CPLD device which manages both the FPGA downloading and the central reset logic for both the cold and the warm reset.

Table 5-29 Reset System

Cold Reset Agents	*Warm Reset Agent*
Front Panel Reset	Front Panel Reset
L/I Power Good	Programmed Reset
CPU Power Monitor	PPMC Reset Request

Table 5-29 Reset System

Cold Reset Agents	*Warm Reset Agent*
Watchdog Timeout	---
CompactPCI SysReset	---

5.5.1 Cold Reset

Until all of the cold reset agents de-assert their requests, all of the logics remain in reset, then the CISP controller start to download all FPGAs from its serial Flash. When the FPGA_Done signal is asserted by the FPGA's, the CPLD central reset deassert all of the different CPU and FPGA reset signal to boot the processor. The PCI RST# signals are released by the FPGA bridges.

FRONT PANEL RESET

The front panel reset-button will initiate the 200 milli-second cold-reset pulse when it is pressed for more than 2 seconds, otherwise it will initiate the warm-reset.

POWER GOOD

The L/I Controller monitor the CompactPCI + 5 Volts, 3.3 Volts, + 12 Volts and - 12 Volts
The onboard power monitor (LTC 1326) check the following Voltage:

- 3.3 Volts LVTTL I/O
- 2.5 Volts L3 core and LVTTL I/O and LDO input
- 1.8 Volts FPGA_Core

The DC/DC switching controller (LTC 1702A) check the following Voltage

- 1.3 Volts CPU_Core

NOTE the LDO 1.5 Volt (HSTL III and L3 Vddq) are not monitored, but the 2.5 Volts power Source is monitored by. the onboard power monitor LTC 1326)

WATCHDOG TIMEOUT

The Real-Time-Clock device provides a fully programmable Watchdog functionality which can generate a cold-reset of the RIOC 4068 board in case of failure.

5.5.2 Warm Reset

The warm reset will not re-load the FPGA logics, but it will just generate a logical reset pulse of 100 micro-seconds (typical), which will reset all the FPGA and the CPU logics.

FRONT PANEL RESET

The front panel reset button will initiate the 100 micro-seconds warm reset pulse when it is pressed for less than 2 seconds, otherwise it will initiate the cold-reset.

PROGRAMMED RESET

The SubI/O local register SUBIO_REG#0 generates a warm reset when its bit [07] is written to "1".

PPMC RESET

The IEEE PPMC specifications adds two reset request signals from the PMC slots, which will generate a warm reset of the RIOC 4068 board.

WARNING The PPMC reset request pulse should be asserted for a minimum of 150 ns.

Chapter 6 CompactPCI Interface

6.1 Introduction

This chapter describes the XPC-to-CompactPCI bridge function incorporated on the RIOC 4068 single board computer. It is implemented in a XILINX VIRTEX 200 KGates FPGA and it occupies a 1 or 2 GBytes address space window over the PowerPC address range.

Table 6-1 CompactPCI Address Mapping

Address	*Description*
0x4000'0000 - 0x7FFF'FFFF	XPC-to-CompactPCI (1 GByte version)
0x4000'0000 - 0xBFFF'FFFF	XPC-to-CompactPCI (2 GBytes version if PCI J3 is disabled)

WARNING

The XPC-PCI_J3 can be disabled to support 2 GBytes over primary bus as CompactPCI. If the RIOC 4068 is plugged in a CompactPCI slot which does not own a PCI_J3 backplane, PCI_J3 bridge must be disabled. This is achieved with PPCMon set command as in table 6-2.

Table 6-2 PPCMon Set Command

Command	*Description*
`ppcmon>set bridge_xpci_ena y <CR>`	Enable the PCI_J3 and therefore CompactPCI occupies only 1 Gbyte.
`ppcmon>set bridge_xpci_ena n <CR>`	Disable the PCI_J3 and therefore CompactPCI occupies 2 Gbytes.

The RIOC 4068 has been designed to be compliant with the latest revisions of the CompactPCI specification (PICMG 2.0 R3.0).

The CompactPCI interface is implemented through “THESEE”, the CES-designed XPC-to-CompactPCI FPGA. The XPC bus is the RIOC’s internal bus which supports the CPU, the memory and the PCI bridges.

To achieve maximum performance, the CompactPCI and the XPC, buses are isolated / buffered through synchronous dual-port SRAM. This memory device is organized in multiple buffers used as “write-buffer” or “read-ahead-buffer”.

This architecture allows the user to build interfaces running at their maximum time-domain frequency.

Table 6-3 Buffer Architecture

Data Size (Bytes)	*Buffer*
8 x 32	PowerPC-to-CompactPCI write-buffer.
4 x 256	DMA controller-to-CompactPCI write or PCI-to- write. Two can be merged to build a “dual-size-buffer” of 512 Bytes.
1 x 256	XPC-to-CompactPCI read-ahead-buffer.
4 x 256	Remote DMA buffer. Data produced by the remote DMA engine.

Table 6-3 Buffer Architecture

Data Size (Bytes)	*Buffer*
4 x 256	CompactPCI-to-XPC write-buffer. Two can be merged to build a “dual-size-buffer” of 512 Bytes. Used by CompactPCI slave transactions.
1 x 256	CompactPCI-to-XPC read-ahead-buffer. Used by CompactPCI slave transactions.

The CompactPCI master interface is implemented through an out-going scatter / gather (64/128 pages of 16 MBytes). Each page is associated to a page descriptor for complete control of the settings of the CompactPCI address, specific control bits for swapping, write-posting, read-prefetch policy, etc.

The CompactPCI slave interface is implemented through an in-coming scatter / gather (page size selectable with 1, 2 or 4 MBytes). Each page is associated to a page descriptor for complete control of the settings of the XPC address, specific control bits for swapping, write-posting, write-protect, read-prefetch policy, etc. The PCI target window can be programmed from 32 to 1024 MBytes.

An autonomous DMA is integrated to implement efficient data transportation between the local system memory, PCI and the CompactPCI. This block-mover provides very high-speed CompactPCI transactions in D32 and D64.

PPCMon and the BSPs for the OS supported by CES, contain high-level functions which handle and initialize the entire DMA register set. The user should not access them directly except for debugging purposes.

6.2 CompactPCI Internal Registers

All internal registers are dual ported between the CompactPCI CONFIG and the XCSR.

Local access to the internal registers are handled through the XCSR bus. Access from the CompactPCI bus is handled through the PCI CONFIG space. The XCSR bus is a high-speed 100 MHz 8-bit bus used to map all general onboard control registers. The XCSR bus master is the SubI/O FPGA (connected on PCI_Local). The XCSR provides 256 32-bit locations per XPC bridge.

If the clocking domain on the PCI side of the bridge is not running, then the XCSR interface is not enabled and read to conf register 0x00 returns 0x00000000 in place of regular Device_ID + Manufacturer_ID (0x4065'10d9). This mechanism allows to confirm if the bridge's PCI clock domain is ready.

Figure 6.1 CompactPCI Local Register

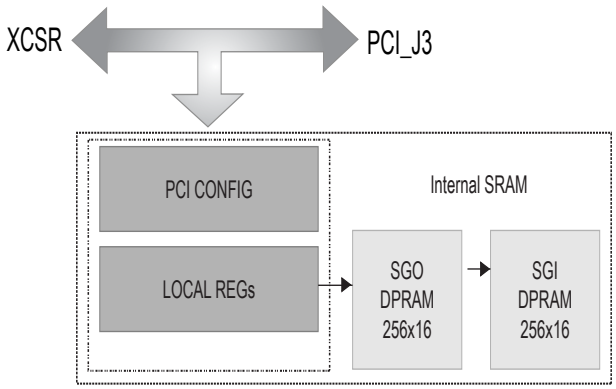


Table 6-4 CompactPCI Internal Registers

CompactPCI Registers	*Local XCSR Mapping (PowerPC Address)*	*CompactPCI Configuration Mapping*
PCI CONF Registers	0xF900'9200 to 0xF900'927F	0x00 to 0x7F
CPCI_LOC_CTL	0xF900'9280	0x80
CPCI_LOC_STA	0xF900'9284	0x84

Table 6-4 CompactPCI Internal Registers

CompactPCI Registers	*Local XCSR Mapping (PowerPC Address)*	*CompactPCI Configuration Mapping*
CPCI_SGADD	0xF900'9288	0x88
CPCI_SGDAT	0xF900'928C	0x8C
CPCI_INTGEN	0xF900'92AC	0xAC
CPCI_SEMA	0xF900'92C0 to 0xF900'92DC	0xC0 to 0xDC
CPCI_RAM	0xF900'92E0 to 0xF900'92FC	0xE0 to 0xFC

NOTE

The access to these registers requires swapping to match the PCI / PowerPC endian conversion. In the following example, PPCMon command should be used.

```
ppcmon>pm.ls f9009200 <CR>
0xf9009200 : 406510d9 -> Result as defined by PCI format
```

6.3 Functional Description

6.3.1 CompactPCI Configuration Registers

COMPACTPCI CONFIGURATION SPACE (STANDARD PCI, ETC.)

The XPC-CompactPCI bridge incorporates the full PCI configuration space. Please refer to PCI specification 2.2 for complete register description.

HOT-SWAP CONTROL AND STATUS REGISTER

This register is accessed through the ECP (Extended Capability Pointer) mechanism:

- Bit 4 of the CompactPCI CSR (Base + 0x02) is asserted to signal the presence of a capabilities list. Capability ID = 0x06 (refer to PCI Specification 2.2).
- Capabilities pointer (Base + 0x34) contains the offset of the first element of the list.

Table 6-5 CompactPCI HS_CSR Mapping

	HS_CSR CPU = 0xF900'9144 & CompactPCI = CONF_Base + 0x44		
Bits	*Name*	*Mode*	*Description*
[23]	INS	R	ENUM# Status - Insertion 1 = ENUM# asserted 0 = not asserted
[22]	EXT	R	ENUM# Status - Extraction 1 = ENUM# asserted 0 = not asserted
[21]	N/A	-	Not Used
[20]	N/A	-	Not Used
[19]	LOO	R/W	Front panel Blue led control
[18]	N/A	-	Not Used
[17]	EIM	R/W	ENUM# Signal Mask 1 = mask signal 0 = enable signal
[16]	N/A	-	Not Used

NOTE

Refer to CompactPCI Hot-Swap specification for a complete description (PICMG 2.1 Rev 2.0).

6.3.2 Local CompactPCI Registers

CPCI_LOC_CTL

The following read / write register provides major XPC-CompactPCI bridge's control bits:

Table 6-6 CPCI_LOC_CTL Mapping

	LOC_CTL CPU = 0xF900'9280 & CompactPCI = CONF_Base + 0x80		
Bits	***Name***	***Mode***	***Description***
[31:25]	N/A	-	Not Used
[24]	EnumEXT_PLS	W	ENUM extraction set command while enumMOD = 01
[23:22]	EnumMOD_EXT	R/W	ENUM extraction signalling mode of operation = 00: ENUM asserted by the front panel switch = 01: ENUM asserted by pgm action from the local CPU = 10: ENUM asserted by the watchdog time-out = 11: ENUM asserted by the watchdog time-out or the pgm action from the local CPU
[21]	ctlpciCOMPREAD	R/W	CompactPCI slave read mode of operation = 0: standard 2.2 delayed read = 1: read compelled (full handshake)
[20]	xpcGEN_PARI	R/W	XPC Parity generation = 0: XPC parity always generated OK = 1: XPC parity generated OK if related PCI cycle have PCI parity OK, otherwise XPC parity are inverted
[19:16]	N/A	R/W	Not Used
[15]	ctlpciDISCTIM	R/W	CompactPCI read slave discard timer. As defined by the PCI specification 2.2 = 0 : 2E16 clocks = 1 : 2E10 clock (short)
[14]	ctlpciCONF_RDY	R/W	When reset, the CompactPCI slave configuration access are continuously retried (bit set after a reset).
[13:12]	N/A	R/W	Not Used
[11:10]	pciARB_MOD	R/W	Internal CompactPCI central arbiter option. These control bits are effective only if the RIOC 4068 is CompactPCI system controller. = 00: standard pipelined + bus parking on XPC-to-CompactPCI bridge. = 01: not pipelined + bus parking on dummy devices assuring the pull-up of active signals. = 10: reserved for debugging = 11: pipelined + bus parking on latest CompactPCI master (PCI 2.2 spec)
[09:08]	N/A	R	Read at 0b00
[07]	CPCI_FLUSH_MOD	R/W	Internal read / write priority resolution for XPC-to-CompactPCI master transactions. = 0: read flushing only over pcidirWB (CPU write transactions) = 1: read flushing over pcidirWB & pcidmaWB (CPU & DMA write transactions)
[06:05]	READ_PFETCH_SIZ	R/W	PCI CompactPCI slave read prefetch size (in 32-bit word) = 00 :6 if PCI32, 8 if PCI64 = 01 :8 if PCI32, 16 if PCI64 = 10 :16 if PCI32, 32 if PCI64 = 11 :32 if PCI32, 64 if PCI64
[04:02]	CPCI_SLVMEM_SIZ	R/W	Define the PCI memory size occupied by the unit = 000: 32 MBytes = 011: 256 MBytes = 001: 64 MBytes = 100: 512 Mbytes = 010: 128 MBytes = 101: 1024 MBytes
[01]	N/A	R/W	Reserved for PCI with 64-bit address. This control bit must never be set.
[00]	pciSIZ64_ENA	R/W	CompactPCI backplane size 64-bit enabled. This control bit is set after power-up reset. Can be reset in special situations.

CPCI_LOC_STA

This read only register provides basic status information on the XPC-to-CompactPCI bridge.

Table 6-7 CPCI_LOC_STA Mapping

	LOC_STA CPU = 0xF900'9284 & CompactPCI = CONF_Base + 0x84		
Bits	***Name***	***Mode***	***Description***
[31:29]	N/A	-	Not Used
[28]	resetctlARMED	R	CompactPCI interface logic ready. This status becomes set while internal initialization sequence is performed.
[27]	HANDswitch	R	Front panel handle switch status. While 0, the switch is open.
[26]	CPCI_CONFENA	R	PCI slave interface enabled. While set the configuration cycles are accepted by the CompactPCI interface.
[25]	CPCI_SYSEN	R	CompactPCI system slot (1 = system controller).
[24:17]	N/A	-	Not Used. Read as 0b0000000
[16:12]	CompactPCI_GA	R	CompactPCI geographical address. = 00001 : Slot 1 = 00010 : Slot 2 = 00011 : Slot 3 .... up to Slot 8
[11]	CPCI_ARB_TO	R/Wc	CompactPCI arbitration time-out flag. Set to 1 if a pciGNT# is issued and no PCI cycle is started after 16 clock cycle. This flag is cleared by writing 1 over it.
[10]	CPCI_READTO	R/Wc	CompactPCI discard timer time-out flag. This flag is cleared by writing 1 over it.
[09]	XPX_WRERR	R/Wc	CompactPCI-to-XPC write error detected. This flag is set in case of XPC error. This flag is cleared by writing 1 over it.
[08]	CPCI_WRERR	R/Wc	XP- to-CompactPCI write error detected. This flag is set in case of CompactPCI error (target abort or master abort). This flag is cleared by writing 1 over it.
[07:00]	N/A	-	Not Used. Read as 0b00000000.

CPCI_SGADD

This register holds the local address pointer for internal SRAM holding the page descriptors associated with the scatter / gather in and out.

Table 6-8 CPCI_SGADD Mapping

	SGADDCPU = 0xF900'9288 & CompactPCI = CONF_Base + 0x88		
Bits	***Name***	***Mode***	***Description***
[31:18]	N/A	-	Not Used
[17:16]	InitRAM_SEL	R/W	This 2-bit field selects the internal SRAM to initialize. = 00 : SGO (XPC-to-CompactPCI scatter / gather) = 01 : SGI (CompactPCI-to-XPC scatter / gather) = 1x : Reserved
[15:09]	N/A	-	Reserved for future extension
[08:00]	sgREG_ADD	R/W	Address pointer to SRAM holding the page descriptor.

CPCI_SGDAT

This 16-bit register allows to read / write the internal SRAM SGO and SGI for initialization.The SRAM location is pointed by the previous register SGADD.

Table 6-9 CPCI_SGDAT Mapping

	SGDATCPU = 0xF900'928C & CompactPCI = CONF_Base + 0x8C		
Bits	***Name***	***Mode***	***Description***
[31:16]	N/A	-	Not Used

Table 6-9 CPCI_SGDAT Mapping

	SGDATCPU = 0xF900'928C & CompactPCI = CONF_Base + 0x8C		
Bits	***Name***	***Mode***	***Description***
[15:00]	sgREG_DATA	R/W	Data register for local SRAM access (read / write)

CPCI_INTGEN

This register allows to generate a programmable interrupt over the CompactPCI.

Table 6-10 CPCI_INTGEN Mapping

	INTGENCPU = 0xF900'92AC & CompactPCI = CONF_Base + 0xAC		
Bits	***Name***	***Mode***	***Description***
[31:06]	N/A	-	Not Used
[05]	CPCIEnable	R/W	CompactPCI interface enable command. The CompactPCI interface is physically disconnected until this control bit is set. (CompactPCI signals are isolated through CMOS quick-switches FST6800)
[04]	RemoteRESET	W	While written to '1', this trigger a local RESET. This control bit can be used to reset a RIOC 4068 across the CompactPCI
[03:00]	INTx_STA	R	Read-back status info of driven CompactPCI. {INTD, INTC, INTB, INTA}
[03]	INTx_GEN_CLR	W	While set, the encoded INTx is cleared.
[02]	INTx_GEN_SET	W	While set, the encoded INTx is set.
[01:00]	INTx_GEN_COD	W	Encoded CompactPCI INTx used with bit [3:2]. = 00 : INTA = 01 : INTB = 10 : INTC = 11 : INTD

WARNING The bits [03:00] do not have the same meaning during read or write.

6.3.3 CompactPCI Semaphores

The RIOC 4068 incorporates eight semaphores / mailbox registers, directly accessible through the configuration space. These resources can be used to implement simple synchronization mechanisms in between CompactPCI boards.

The reservation mechanism sets the owner bit [31] under read access while the semaphore is currently not reserved. An external agent (i.e. another RIOC 4068) can read the semaphore location and test the bit [31] read. If the read result bit[31] is set, then the semaphore is owned by another agent. If the bit [31] is cleared, the reservation is acquired and the hardware will set the owner bit [31], assuring the ownership reservation. The semaphores locations must be accessed only in D32 (no Byte or half-word access).

A mailbox interrupt, bit [30] is associated with each semaphore, allowing to signal an interrupt to the local PowerPC CPU. The interrupt is cleared by writing into the semaphore location from the local CPU.

The eight RAM locations mapped from 0xE0 to 0xFC are provided for external signature. They act as regular SRAM without any correlation with the semaphores. There is no interrupt generation associated with these RAM locations.

Table 6-11 Semaphores mapping

Semaphore	***PCI Configuration Space***	***Local CPU Address***
SEMA_0	0xC0	0xF90092C0
SEMA_1	0xC4	0xF90092C4
SEMA_2	0xC8	0xF90092C8
SEMA_3	0xCC	0xF90092CC
SEMA_4	0xD0	0xF90092D0

Table 6-11 Semaphores mapping

Semaphore	*PCI Configuration Space*	*Local CPU Address*
SEMA_5	0xD4	0xF90092D4
SEMA_6	0xD8	0xF90092D8
SEMA_7	0xDC	0xF90092DC
RAM[00:07]	0xE0 to 0xFC	0xF90092E0 to 0x90092FC

CPCI_SEMA

These 32-bit register implements the eight semaphores control.

Table 6-12 CPCI_SEMA Mapping

	SEMACPU = 0xF900'92C0 to 0xF900'92EC & CompactPCI = CONF_Base + 0xC0 to DC		
Bits	*Name*	*Mode*	*Description*
[31]	OWNER	R/W	Owner bit. This bit is used to implement the reservation mechanism. If read while owner = 0, then owner is automatically set after the read. The owner bit must be cleared manually by the owner.
[30]	MBINT	R/W	This bit allows to generate the local mailbox interrupt.
[29:24]	Reserved	R	Not Used. Read with 0b000000
[23:16]	MB_INT[07:00]	R	Mailbox interrupt status. This 8-bit read only field gives the status info about the 8 mailboxes related to the 8 semaphores.
[15:00]	semaDATA	R/W	Semaphore data field. Used to sign the semaphore message.

CPCI_RAM

These 32-bit register implements the eight semaphores control.

Table 6-13 CPCI_RAM Mapping

	RAMCPU = 0xF900'92E0 to 0xF900'92FC & CompactPCI = CONF_Base + 0xE0 to FC		
Bits	*Name*	*Mode*	*Description*
[31:16]	Reserved	R	Shadow read of SEMA registers.
[15:00]	ramDATA	R/W	Standard 16-bit RAM location.

6.4 CompactPCI Target Port (Slave)

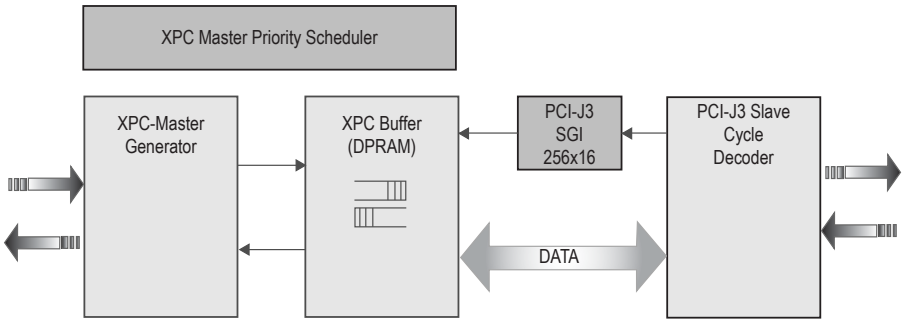
6.4.1 Functional Description

The CompactPCI slave decoding uses:

- Base Address Register (BAR)
- Memory size (programmable from 32 to 1024 MBytes)

The CompactPCI-to-XPC write are completely decoupled through write-buffers (all writes are posted). The XPC-to-CompactPCI read are isolated with a read-ahead buffer and can be programmed as read delayed (PCI Rev 2.2) or fully compelled (PCI 2.1). These two mechanisms allow to increase the general throughput of the CompactPCI bridge. It also offers a better use of XPC-bus by providing a full speed XPC-master interfacing.

Figure 6.2 CompactPCI Slave



6.4.2 CompactPCI XPC Master Cycle Scheduler

The XPC master scheduler is directly linked to the XPC arbitration request process. It continuously evaluates the local transactions pending following a pre-determined priority. In case of CompactPCI-to-XPC read pending and rflush active, the scheduler first serves the xpcdirWBs to empty them before executing the xpcdirRD. The XPC arbitration implements four priority levels (4 down to 1) to minimize the latency.

Table 6-14 XPC Buffer

PRI_level	*Buffers*	*Description*	*XPC Arbitration*
4	xpcdirWB	CompactPCI-to-XPC read with flush pending. Serve first the xpcdirWB until no more are available.	xpcREQ = 11
3	xpcdirRD	CompactPCI-to-XPC read pending	xpcREQ = 11
2	xpcdirWB	CompactPCI-to-XPC write direct buffer pending	xpcREQ = 10

6.4.3 CompactPCI A32 Scatter / Gather IN

To map A32 CompactPCI slave decoded window over the XPC bus, the missing information is extracted from an in-coming scatter / gather. The in-coming scatter / gather is organized in 256 x 16, which provides 256 entries. The page size is conditioned by the PCI slave window size programmed with control bits “CPCI_SLVMEM_SIZ” located in the CPCI_LOC_CTL register.

- 1 MByte, if the slave window is programmed to 32 to 256 MBytes
- 4 MBytes, if the slave window is programmed to 512 to 1024 MBytes

The XPC address translation is executed during the CompactPCI slave access. The SGI initialization is handled through the internal registers CompactPCI_SGADD and CompactPCI_SGDAT (which are also accessible from the CompactPCI CR/CSR space). The SGI memory, which stores the 256 page entries, is built with fast dual-port SRAM.

Each page entry is associated with a direct mapped SGI page descriptor. The page descriptor is built over 16-bits which provide support for the XPC cycle built.

Table 6-15 CompactPCI SGI Page Descriptor

Bits	*Name*	*Description*
[15:08]	xpcADD[31:24]	XPC top address remapping
[07:04]	xpcADD[23:20]	XPC top address remapping
[03:02]	TDST	XPC target destination = 00 : system memory (PowerPC bus) = 01 : local PCI bridge = 10 : extension CompactPCI = 11 : do not use
[01]	xpcSNOOP	This bit is conditioned by xpcDST[1:0] If xpcDST[1:0] = 00 : enable snooping in PowerPC target If xpcDST[1:0] /= 00 : (=0) PCI memory space (=1) PCI I/O space
[00]	xpcRFLUSH	This bit enable read with flush

NOTE

The SGI in XPC-to-CompactPCI bridge is initialized with a remapping schema full filling the general XPC 32-bit mapping.

Since the SGI memory can also be initialized / modified from the CompactPCI CONFIG space, it is possible to control the CompactPCI-to-XPC remapping completely from the CompactPCI side.

6.4.4 CompactPCI Configuration Cycles

The CompactPCI bridge implements a specific mechanism to support slave config space when it is plugged in the system controller slot (Slot 1). This mechanism allows access to the system controller's configuration space from the peripheral slot, especially to the top area (0x80 -0xFF) holding the local registers.

Table 6-16 CPCI Configuration

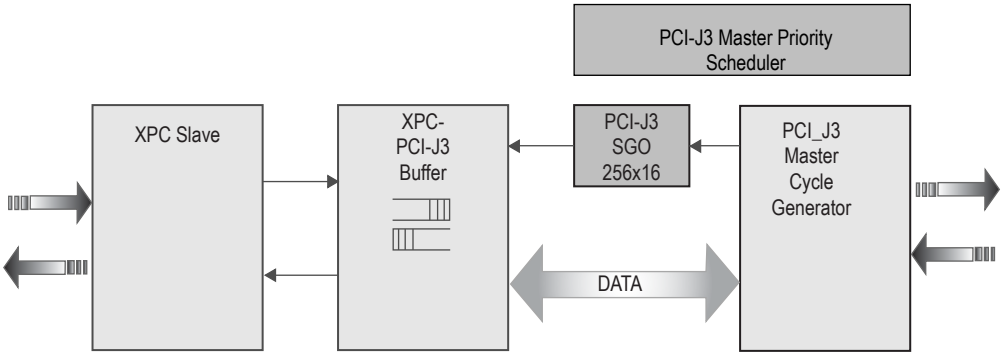
System Controller	*Description*
Slot 2 to Slot 8	Standard PCI selection with IDSEL selection (IDSEL extracted from pciAD[31:24])
Slot 1	Selection with pciAD23 (virtual IDSEL extracted from pciAD23)

6.5 CompactPCI Initiator Port (Master)

6.5.1 Functional Description

The PCI master interface handles the XPC-to-CompactPCI transparent transactions and the embedded DMA transactions. For both transaction controls, the PCI interface is isolated with an out-going scatter / gather (SGO) tailored in 16 MBytes page size.

Figure 6.3 PCI Master



The XPC-to-CompactPCI write are completely decoupled through write-buffers (all writes are posted). The XPC-to-CompactPCI read are isolated with a read-ahead buffer, these two mechanisms allow to increase the general throughput of the CompactPCI bridge. It also provides a better use of XPC-bus by providing a full speed XPC-slave interfacing.

6.5.2 CompactPCI A32 Scatter / Gather OUT

The out-going scatter / gather is organized in 512x16 which provides 128 entries of 16 MBytes. The CompactPCI address translation is executed during the PCI master access (and not during the XPC slave access).

The SGO initialization is handled through the internal registers CompactPCI_SGADD and CompactPCI_SGDAT (which are also accessible from the PCI CR/CSR configuration space). The SGO memory, which stores the entries, is built with fast dual-port SRAM, allowing fast translation.

Table 6-17 CompactPCI SGO Page Descriptor

Bits	*Name*	*Description*
[63:40]	N/A	Not Used
[39:32]	pciADD[63:56]	PCI top address (reserved for PCI A64)
[31:16]	pciADD[47:32]	PCI top address (reserved for PCI A64)

Table 6-17 CompactPCI SGO Page Descriptor

Bits	*Name*	*Description*
[15:08]	pciADD[31:24]	PCI top address
[07:05]	A_Type	Address space field. = 000 : memory A32 = 001 : memory A64 = 010 : I/O = 011 : config = Others : reserved
[04:02]	D_Type	Data type encoding for PCI memory space = 000 : Std read / write = 001 : read multiple / write = 010 : read line/ write invalidate = Others : reserved
[01]	P_Type	PCI_D64 enable. If possible use PCI D64 transactions
[00]	N/A	Not Used

6.5.3 XPC-to-CompactPCI Configuration Cycles

The PCI Configuration Cycles are generated through a 16 MBytes window programmed with AType = "011". Only the CompactPCI system slot can perform configuration cycles. The A23-00 address field is assigned as follows:

Figure 6.4 Layout of Config_Address Register

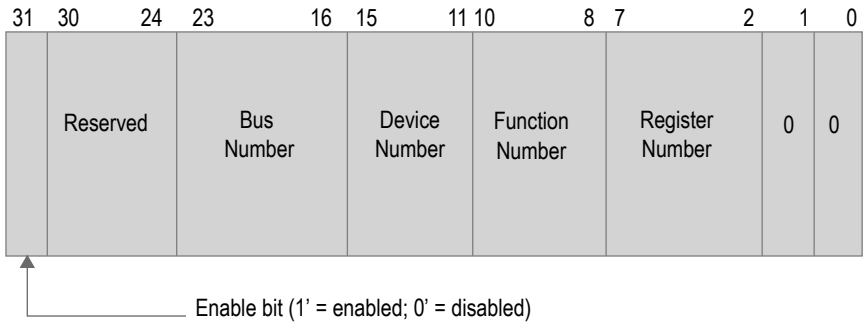
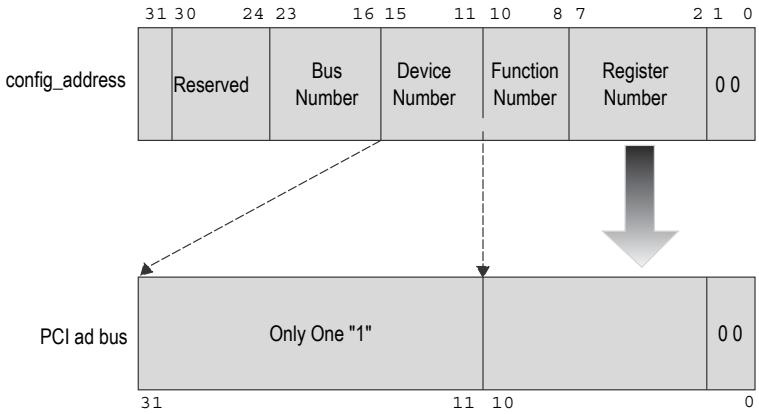


Table 6-18 CompactPCI Configuration Cycle Encoding

Address bit	*Name*	*Description*
[31:24]	N/A	Not used in the 16 MBytes page
[23:16]	Bus Number	8-bit field selecting Type_0 or Type_1 configuration cycle type. If Bus_Number = 0x00, type_0 is issued with: Device_ID = 00000 : reserved 00001 : Slot 1 ! special 00010 : Slot 2 00011 : Slot 3 ....
[15:11]	Device_ID	Device number
[10:09]	Function_ID	Function number
[08:02]	Register_ID	Register number
[01:00]		Fixed to “00”

NOTE The Type_0 cycle with Device_ID=00001 is used to access the configuration registers of the system controller.

Figure 6.5 PCI Configuration Cycle Encoding



6.5.4 CompactPCI Bridge Interrupts

The related CompactPCI interrupts are all routed to the System Interrupt Controller as follows:

Table 6-19 CompactPCI Interrupts

CompactPCI Interrupts	*SIC ACTcode*	*Description*
CompactPCI_INTA	0x20	CompactPCI INTA, or function of all INTA coming from CompactPCI backplane
CompactPCI_INTB	0x1F	Idem for CompactPCI INTB
CompactPCI_INTC	0x1E	Idem for CompactPCI INTC
CompactPCI_INTD	0x1D	Idem for CompactPCI INTD
CompactPCI_SERR	0x1B	Idem for CompactPCI SERR
CompactPCI_PERR	0x1C	Not Used
CompactPCI_ENUM	0x1A	Allows to detect CompactPCI extraction
CompactPCI_MBIRQ	0x2D	CompactPCI mailbox interrupts associated with the semaphores

Please refer to the chapter concerning SIC interrupt controller.

NOTE On CompactPCI backplane, the interrupt INTA - INTB - INTC - INTD are reassigned (rolled) on every slot. Please refer to CompactPCI specification.

6.5.5 PPCMon Support for CompactPCI

The onboard monitor PPCMon allows the configuration and the access of the CompactPCI interface.

PPCMON "SET CPCI" COMMAND

The PPCMon command "set bridge_cpci" allows the configuration of the CompactPCI interface at power-up. Any changes are executed immediately. Information kept by this command is stored in a non-volatile memory. The following subsets are used by the command:

```
ppcmon> set bridge_cpci <CR>
```

Table 6-20 Set XPCI Option

subset option	*Description*
bridge_cpci_ena	Enables the CompactPCI Bridge Interface. This option must be enabled only if a PCI_J3 backplane is installed. If disabled the XPC-to-CompactPCI bridge will occupy 2 GBytes **y** : CompactPCI enabled **n** : CompactPCI disabled
bridge_cpci_s64	Defines the CompactPCI data width **y** : CompactPCI 64-bit **n** : CompactPCI 32-bit
bridge_cpci_a64	Defines the CompactPCI addressinf **y** : CompactPCI 64-bit address enabled **n** : CompactPCI64-bit address disabled
bridge_cpci_size	Defines the PCI target window size. Defined as: **32**, **64**, **128**, **256**, **512,1024**: Size in MBytes
bridge_cpci_prefetch	Sets block size of PCI target read prefetch. Defined as: **12**, **32**, **64**, **128** : Bytes if D32 **32**, **64**, **128**, **256** : Bytes if D64
bridge_cpci_flush	Defines the XPC-CompactPCI flush policy. Defined as: **y** : Read flush only over direct PowerPC to CompactPCI writes **n** : Read flush over PowerPC and DMA writes
bridge_cpci_arb	CompactPCI central arbitration mode **0, 1, 2, 3** : Arbitration mode (refer to register CPCI_LOC_CTL)
bridge_cpci_timer	Defines the disconnect timer value. If the read are defined with “bridge_xpci_rcmpld = y“, the short value should be selected **2E10** : short value **2E16** : long value, PCI 2.2 with uncompelled read
bridge_cpci_parity	Defines the XPC-to-CompactPCI parity reporting policy **y** : parity error over PCI is reported to XPC **n** : parity error over PCI is not reported over XPC
bridge_cpci_rcmpld	Defines the read policy **y** : read compelled (PCI 2.1) **n** : read uncompelled (PCI 2.2)
bridge_cpci_enum	Compact PCI ENUM signal assertion on following selected condition: switch, prog, watchdog, disable.

NOTE

Many of above options are directly related to bits located in the CPCI_LOC_CTL register.

PPCMON ACCESS COMMANDS

The PPCMon command pc, dc, tc,... allows to directly access the CompactPCI without requiring any SGO initialization. All of the commands have the same synthax, as follows:

```
ppcmon>pc.s addr data mode <CR>
```

with following signification:

`s`	`b, h, l`	Size as byte, halfword, long
`addr`		Address in hexadecimal value (4,6 or 8 digit)
`data`		Data in hexadecimal value (2, 4 or 8 digit)
`mode`		Mode in alphanumeric as mem or io

Table 6-21 Basic CompactPCI Access Debugging Command

Command	*Description*
pc	Patch CompactPCI. Dynamic read / write access over CompactPCI pc.l 8203000 `0 ? mem <CR>` ; CompactPCI read long word `pc.b 82030000 12 mem <CR>` ; CompactPCI Memory write
dc	Display CompactPCI. Display value of CompactPCI location `dc.l 84030000 io<CR>` ; Display 80 consecutive location `dc.l 84030000..84031000 <CR>` ; Display specified range

PPCMon uses only the first CompactPCI page 0x4000'0000-0x40FF'FFFF. The scatter / gather page descriptor is reloaded continuously.

NOTE For complete PPCMon command list, please refer to the PPCMon User's Manual (3317C/UI).

PPCMON STATUS & CONFIG COMMANDS

The PPCMon command "xpci config" displays the current setting of the CompactPCI interface. It can provide a quick debugging status of the onboard XPC-to-CompactPCI bridge.

```
ppcmon>cpci config <CR>

CPCI interface [enabled]
System Controller
Slave      : A32 Size = 32 Mbytes
             A32 Base = 0x80000000
             A64 Base = 0x00000000'00000000  [disabled]
```

The PPCMon command "`cpci show`" prints out the CompactPCI physical allocation.

```
ppcmon>cpci show <CR>

CPCI interface [enabled]
   System Controller
   Slave       : A32 Size = 32 Mbytes
                 A32 Base = 0x40000000
                 A64 Base = 0x00000000'00000000  [disabled]

+----+-----------------------------------------+
|              PCI DEVICES  [tree 8 ]          |
+----+------+------+----+----+----+----+----+----+
|idx | vid  | did  |slot|func|bus |b_id|ivec|pmc |
+----+------+------+----+----+----+----+----+----+
|  0 | 10d9 | 4065 | 00 | 00 | 00 |  0 | 00 |  x |
|  1 | 10d9 | 4065 | 01 | 00 | 00 |  0 | 00 |  1 |
|  2 | 10d9 | 4064 | 04 | 00 | 00 |  0 | 00 |  4 |
+----+------+------+----+----+----+----+----+----+


PPC_Mon>cpci show dev 4

Single function device found in slot : 0x04   bus : 0x00

| vid  | did  | type |   addr   |   size   | space| bar  |
+------+------+------+----------+----------+------+------+
| 10d9 | 4064 |  D   | 43000000 | 00100000 | MEM  |  0   |
                     | b0000000 | 00001000 | IO   |  1   |
                     | 42000000 | 01000000 | MEM  |  2   |
                     | 43100000 | 00100000 | MEM  |  3   |
                     | 43200000 | 00100000 | PROM |
```

The PPCMon command "`diag cpci`" also scans the CompactPCI backplane, executes the PCI configuration (physical memory allocation) and prints out current allocations.

Chapter 7
Rear Panel Bus PCI (PCI#2)

7.1 Introduction

This chapter describes the XPC-to-PCI_J3 bridge function incorporated on the RIOC 4068 CompactPCI single board computer. It is implemented in a XILINX VIRTEX 150 KGates FPGA and it occupies a 1 GByte address space window over the PowerPC address range.

Main features:
- PCI64 / PCI32 33 MHz
- Address remapping to/from XPC Bus
- Fully programmable in-going & out-going scatter / gather
- Hot-Swap capability similar to CompactPCI
- Interrupt dispatching to / from each slot

Table 7-1 PCI_J3 Address Mapping

Address	*Description*
0x8000'0000 - 0xBFFF'FFFF	PCI_J3 (PCI Extension over CompactPCI J3)

WARNING

The XPC-PCI_J3 can be disabled to support 2 GBytes over primary bus as CompactPCI. If the RIOC 4068 is plugged in a CompactPCI slot which does not own a PCI_J3 backplane, PCI_J3 bridge must be disabled. This is achieved with PPCMon set command as in table 7-2.

Table 7-2 PPCMon Set Command

Command	*Description*
`ppcmon>set bridge_xpci_ena y <CR>`	Enable the PCI_J3 and therefore CompactPCI occupies only 1 GByte.
`ppcmon>set bridge_xpci_ena n <CR>`	Disable the PCI_J3 and therefore CompactPCI occupies 2 GBytes.

7.1.1 CompactPCI_J3

The VME P0 connector and the CompactPCI J3 connector are identical. Both supply 95 signals and two rows of GND (only 1 row is implemented on the RIO3 8064). These connectors are user-free on both standard and they are used as a PCI extension for the RIOC 4068. A 64-bit PCI is supplied by the XPC-to-PCI_J3 bridge.

The interconnection is built with a passive backplane (CES number BPA 6413 and BPA 6414) which is plugged on the back of the CompactPCI backplane (similar to VSB). Two and three slot versions are available.

The XPC-to-PCI_J3 bridge allows the user to interconnect up to three CPUs (RIOC 4068) or a CPU with two PMC extension boards (PEB 6418).

NOTE

CASTOR is the first XPC-PCI_J3 implementation. It provides minimal functionality based on a 64-bit PCI, since the number of available pins on J3 is a limiting factor. A future extension based on PCI32 will supply an extension over 8-slots with Hot-Swap capability.

MULTIPLE CPU INTERCONNECTION

The PCI_J3 extension (BPA 6414) can be used to link up to three RIOC 4068 boards. In this case the three PCI configuration spaces are available and only one CPU must be dedicated to initialize the sub-system. A system controller information is supplied by the backplane to enable the PCI arbiter and a status is provided in the PCI_J3_LOC_STA register. The system controller (PCI arbiter) is the right-most board relative to the PCI_J3 backplane.

CPU + PEB INTERCONNECTION

The PCI_J3 extension can also be used to expand the PMC capability. A system controller information is supplied by the backplane to enable the PCI arbiter and a status is provided in the PCI_J3_LOC_STA register. The system controller (PCI arbiter) is the right-most board relative to the PCI_J3 backplane.

NOTE

- PCI_J3 Master Automatic detection
- SW400-2 select for single or dual clock BPA 6414

7.1.2 XPC-to-PCI_J3 Buffer

To achieve maximum performance, the PCI_J3 and the XPC bus are isolated / buffered through synchronous dual-port SRAM. This memory devices is organized in multiple buffers used as 'write-buffer' or 'read-ahead buffer'. This architecture allows to build an interface running at its maximum time-domain frequency. Every buffer is made of one 64-bit word address/ header and N data.

Table 7-3 XPC Buffer

Buffer name	*Number*	*Data Size (Bytes)*	*Function*
pcidirWB	8	32	PowerPC-to-PCI_J3 write buffer
pcidmaWB	4	256	DMA controller to PCI_J3 write or PCI to PCI write. Two can be merged to build a 'dual-size buffer' of 512 bytes.
pcidirRD	1	256	XPC-to-PCI read ahead buffer
xpcdmaWB	4	256	Remote DMA buffer. Data produced by the remote DMA engine. Not used in XPC-to-PCI_J3 bridge.
xpcdirWB	4	256	PCI_J3-to-XPC write buffer. Two can be merged to build a 'dual-size buffer' of 512 bytes. Used by PCI_J3 slave transactions.
xpcdirRD	1	256	PCI_J3 to XPC READ ahead buffer. Used by PCI slave transactions.

7.2 PCI_J3 Internal Registers

All internal registers are dual ported between the PCI_J3 CONFIG and the XCSR.

Local access to the internal registers are handled through the XCSR bus. Access from the PCI_J3 bus is handled through the PCI CONFIG space. The XCSR bus is a high-speed 100 MHz 8-bit bus used to map all general onboard control registers. The XCSR bus master is the SubI/O FPGA (connected on PCI_Local). The XCSR provides 256 32-bit locations per XPC bridge.

If the clocking domain on the PCI side of the bridge is not running, then the XCSR interface is not enabled and read to conf register 0x00 returns 0x00000000 in place of regular Device_ID + Manufacturer_ID (0x6065'10d9). This mechanism allows to confirm if the bridge's PCI clock domain is ready.

Figure 7.1 PCI_J3 Local Register

Table 7-4 PCI_J3 Internal Registers

PCI_J3 Registers	*Local XCSR Mapping (PowerPC Address)*	*PCI_J3 Configuration Mapping*
PCI CONF Registers	0xF900'9100 to 0xF900'917F	0x00 to 0x7F
PCI_J3_LOC_CTL	0xF900'9180	0x80
PCI_J3_LOC_STA	0xF900'9184	0x84
PCI_J3_SGADD	0xF900'9188	0x88
PCI_J3_SGDAT	0xF900'918C	0x8C
PCI_J3_INT_ENA	0xF900'91A0	0xA0
PCI_J3_INT_REQ	0xF900'91A4	0xA4
PCI_J3_INT_GEN	0xF900'91AC	0xAC

NOTE

The access to these registers requires swapping to match the PCI / PowerPC endian conversion. In the following example, PPCMon command should be used:

```
ppcmon>pm.ls f9009100 <CR>
0xf9009100 : 606510d9 ->Result as defined by PCI format
```

7.3 Functional Description

7.3.1 PCI_J3 Configuration Registers

PCI_J3 CONFIGURATION SPACE (STANDARD PCI, ETC.)

The XPC-PCI_J3 bridge incorporates the full PCI configuration space. Please refer to PCI specification 2.2 for complete register description.

HOT-SWAP CONTROL AND STATUS REGISTER

This register is accessed through the ECP (Extended Capability Pointer) mechanism:

- Bit 4 of the PCI_J3 CSR (Base + 0x02) is asserted to signal the presence of a capabilities list. Capability ID = 0x06 (refer to PCI Specification 2.2).
- Capabilities pointer (Base + 0x34) contains the offset of the first element of the list.

Table 7-5 PCI_J3 HS_CSR Mapping

	HS_CSRCPU = 0xF900'9144 & PCI_J3 = CONF_Base + 0x44		
Bits	*Name*	*Mode*	*Description*
[23]	INS	R	ENUM# Status - Insertion 1 = ENUM# asserted 0 = not asserted
[22]	EXT	R	ENUM# Status - Extraction 1 = ENUM# asserted 0 = not asserted
[21]	N/A	-	Not Used
[20]	N/A	-	Not Used
[19]	LOO	R/W	Not Used
[18]	N/A	-	Not Used
[17]	EIM	R/W	ENUM# Signal Mask: 1 = mask signal 0 = enable signal
[16]	N/A	-	Not Used

7.3.2 Local PCI_J3 Registers

PCI_J3_LOC_CTL

The following read / write register provides major XPC-PCI bridge's control bits:

Table 7-6 PCI_J3_LOC_CTL Mapping

	LOC_CTLCPU = 0xF900'9180 & PCI_J3 = CONF_Base + 0x80		
Bits	***Name***	***Mode***	***Description***
[31:25]	N/A	-	Not Used
[24]	EnumEXT_PLS	W	ENUM extraction set command while enumMOD = 01.
[23:22]	EnumMOD_EXT	R/W	ENUM extraction signalling mode of operation: = 01: ENUM asserted by pgm action from the local CPU. Others: Reserved
[21]	ctlpciCOMPREAD	R/W	PCI_J3 slave read mode of operation = 0: Standard 2.2 delayed read = 1: Read compelled (full handshake)
[20]	xpcGEN_PARI	R/W	XPC parity generation. = 0: XPC parity always generated OK = 1: XPC parity generated OK if related PCI cycle have PCI parity OK, otherwise XPC parity are inverted
[19:16]	N/A	-	Not Used
[15]	ctlpciDISCTIM	R/W	PCI_J3 read slave discard timer. As defined by the PCI specification 2.2 = 0: 2E16 clocks = 1: 2E10 clock (short)
[14]	ctlpciCONF_RDY	R/W	When reset, the PCI_J3 slave configuration access are continuously retried (bit set after a reset).
[13:12]	N/A	-	Not Used
[11:10]	pciARB_MOD	R/W	Internal PCI_J3 central arbiter option. These control bits are effective only if the RIOC 4068 is PCI_J3 System controller. = 00: standard pipelined + bus parking on XPC-PCI_J3 bridge = 01: not pipelined + bus parking on dummy devices assuring the pull-up of active signals = 10: reserved for debugging = 11: pipelined + bus parking on latest PCI_J3 master (PCI 2.2 spec)
[09:08]	N/A	-	Not Used
[07]	PCI_J3_FLUSH_MOD	R/W	Internal read / write priority resolution for XPC to PCI_J3 master transactions. = 0: read flushing only over pcidirWB (CPU write transactions) = 1: read flushing over pcidirWB & pcidmaWB (CPU & DMA write transactions)
[06:05]	READ_PFETCH_SIZ	R/W	PCI _J3 slave read prefetch size (in 32-bit word) = 00: 6 if PCI32, 8 if PCI64 = 01: 8 if PCI32, 16 if PCI64 = 10: 16 if PCI32, 32 if PCI64 = 11: 32 if PCI32, 64 if PCI64
[04:02]	PCIJ3_SLVMEM_SIZ	R/W	Define the PCI memory size occupied by the unit. = 000: 32 MBytes = 011: 256 MBytes = 001: 64 MBytes = 100: 512 Mbytes = 010: 128 MBytes = 101: 1024 MBytes
[01]	N/A	-	Not Used
[00]	pciSIZ64_ENA	R/W	PCI_J3 backplane size 64-bit enabled. This control bit is set after power-up reset. Can be reset in special situations.

PCI_J3_LOC_STA

This read only register provides basic status information on the XPC to PCI_J3 bridge.

Table 7-7 PCI_J3_LOC_STA Mapping

	LOC_STACPU = 0xF900'9184 & PCI_J3 = CONF_Base + 0x84		
Bits	***Name***	***Mode***	***Description***
[31]	N/A	-	Not Used

Table 7-7 PCI_J3_LOC_STA Mapping

	LOC_STACPU = 0xF900'9184 & PCI_J3 = CONF_Base + 0x84		
Bits	***Name***	***Mode***	***Description***
[30]	J3_Backplane_Present	R	Reserved
[29]	csrJ3_SERIAL_RUN	R	While set, serial status bus is running, therefore PCI_J3_HEALTHY (x) and LOC_INTP_REQ register are valid.
[28]	resetctlARMED	R	PCI_J3 interface logic ready. This status becomes set while internal initialization sequence is performed.
[27]	N/A	-	Not Used
[26]	PCI_J3_CONFENA	R	PCI slave interface enabled. While set the configuration cycles are accepted by the PCI_J3 interface.
[25]	PCI_J3/_SYSEN	R	PCI_J3 system slot (1 = system controller).
[24:20]	N/A	-	Not Used
[19]	PCI_J3_HEALYTHY3	R	PCI_J3 Slot_3 HEALTHY status bit
[18]	PCI_J3_HEALYTHY2	R	PCI_J3 Slot_2 HEALTHY status bit
[17]	PCI_J3_HEALYTHY1	R	PCI_J3 Slot_1 HEALTHY status bit
[16:12]	PCI_J3_GA	R	PCI_J3 geographical address. = 00001 : Slot 1 = 00010 : Slot 2 = 00011 : Slot 3
[11]	PCI_J3_ARB_TO	R/Wc	PCI_J3 arbitration time-out flag. Set to 1 if a pciGNT# is issued and no PCI cycle is started after 16 clock cycle. This flag is cleared by writing '1' over it.
[10]	PCI_J3_READTO	R/Wc	PCI discard timer time-out flag. This flag is cleared by writing 1 over it.
[09]	XPX_WRERR	R/Wc	PCI_J3- to-XPC write error detected. This flag is set in case of XPC error. This flag is cleared by writing 1 over it.
[08]	PCI_J3_WRERR	R/Wc	XPC-to-PCI_J3 write error detected. This flag is set in case of PCI_J3 error (target abort or master abort). This flag is cleared by writing 1 over it.
[07:00]	N/A	-	Not Used

PCI_J3_SGADD

This register holds the local address pointer for internal SRAM, which holds the page descriptors associated with the scatter / gather in and out.

Table 7-8 PCI_J3_SGADD Mapping

	SGADDCPU = 0xF900'9188 & PCI_J3 = CONF_Base + 0x88		
Bits	***Name***	***Mode***	***Description***
[31:18]	N/A	-	Not Used
[17:16]	InitRAM_SEL	R/W	This 2-bit field selects the internal SRAM to initialize. = 00 : SGO (XPC-to-PCI scatter / gather) = 01 : SGI (PCI-to-XPC scatter / gather) = 1x : Reserved
[15:10]	N/A	-	Reserved for future extension
[09:00]	sgREG_ADD	R/W	Address pointer to SRAM holding the page descriptor.

PCI_J3_SGDAT

This 16-bit register allows to read / write the internal SRAM SGO and SGI for initialization.The SRAM location is pointed by the previous register SGADD.

Table 7-9 PCI_J3_SGDAT Mapping

	SGDATCPU = 0xF900'918C & PCI_J3 = CONF_Base + 0x8C		
Bits	***Name***	***Mode***	***Description***
[31:16]	N/A	-	Not Used
[15:00]	sgREG_DATA	R/W	Data register for local SRAM access (read / write)

PCI_J3_INT_ENA

This register holds the enable for the interrupts coming from the PCI_J3 backplane. These control bits must be set individually to enable PCI_J3 interrupt schema connected to the local System Interrupt Controller.

Table 7-10 PCI_J3_INT_ENA Mapping

	INTENACPU = 0xF900'91A0 & PCI_J3 = CONF_Base + 0xA0		
Bits	***Name***	***Mode***	***Description***
[31:16]	N/A	-	Not Used
[15]	Slot_3_INTD_MSK	-	Interrupt mask for INTD Slot#3. While set, the interrupt source is masked.
[14]	Slot_3_INTC_MSK	-	Interrupt mask for INTC Slot#3
[13]	Slot_3_INTB_MSK	-	Interrupt mask for INTB Slot#3
[12]	Slot_3_INTA_MSK	-	Interrupt mask for INTA Slot#3
[11]	Slot_2_INTD_MSK	-	Interrupt mask for INTD Slot#3
[10]	Slot_2_INTC_MSK	-	Interrupt mask for INTC Slot#3
[09]	Slot_2_INTB_MSK	-	Interrupt mask for INTB Slot#3
[08]	Slot_2_INTA_MSK	-	Interrupt mask for INTA Slot#3
[07:03]	N/A	-	Not Used
[02]	Slot_3_INTENA	R/W	When set the interrupt(s) request coming from Slot#3 issue interrupt to the SIC
[01]	Slot_2_INTENA	R/W	When set the interrupt(s) request coming from Slot#2 issue interrupt to the SIC
[00]	Slot_1_INTENA	R/W	When set the interrupt(s) request coming from Slot#3 issue interrupt to the SIC

PCI_J3_INT_REQ

This register provides the status information for all PCI_J3 interrupts. It is a read only register and should be used to determine the interrupt's source.

Table 7-11 PCI_J3_INTREQ Mapping

	INTREQCPU = 0xF900'91A4 & PCI_J3 = CONF_Base + 0xA4		
Bits	***Name***	***Mode***	***Description***
[31:27]	N/A	-	Reserved for extension INTD
[26]	Slot_3_INTD	R	INTD# IRQ coming from Slot 3
[25]	Slot_2_INTD	R	INTD# IRQ coming from Slot 2
[24]	Slot_1_INTD	R	INTD# IRQ coming from Slot 1
[23:19]	N/A	-	Reserved for extension INTC
[18:16]	Slot_n_INTC	R	INTC# IRQ coming from Slot 3, Slot 2, Slot 1
[15:11]	N/A	-	Reserved for extension INTB
[10:08]	Slot_n_INTB	R	INTB# IRQ coming from Slot 3, Slot 2, Slot 1
[07:03]	N/A	-	Reserved for extension INTA
[02:00]	Slot_n_INTA	R	INTA# IRQ coming from Slot 3, Slot 2, Slot 1

PCI_J3_INTGEN

This register allows to generate a programmable interrupt over the PCI_J3 backplane. It is mainly used for test purposes.

Table 7-12 PCI_J3_INTGEN Mapping

	INTGENCPU = 0xF900'91AC & PCI_J3 = CONF_Base + 0xAC		
Bits	***Name***	***Mode***	***Description***
[31:6]	N/A	-	Reserved for extension INTD
[05]	PCI_J3_IFENA	R/W	PCI_J3 interface enable command. Until this control bit is set, the board is physically disconnected from the PCI_J3 backplane.
[04]	N/A	-	Not Used
[03:00]	INTx_STA	R	Read-back status info of driven PCI_J3_INTx. [0] : INTA, [10] : INTB, etc.
[03]	INTx_GEN_CLR	W	While set, the encoded INTx is cleared.
[02]	INTx_GEN_SET	W	While set, the encoded INTx is set.
[01:00]	INTx_GEN_COD	W	Encoded PCI_J3 INTx used with bit [3:2]. = 00 : INTA = 01 : INTB = 10 : INTC = 11 : INTD

WARNING The bit [03:00] does not have the same meaning during read or write.

7.4 PCI_J3 Target Port (Slave)

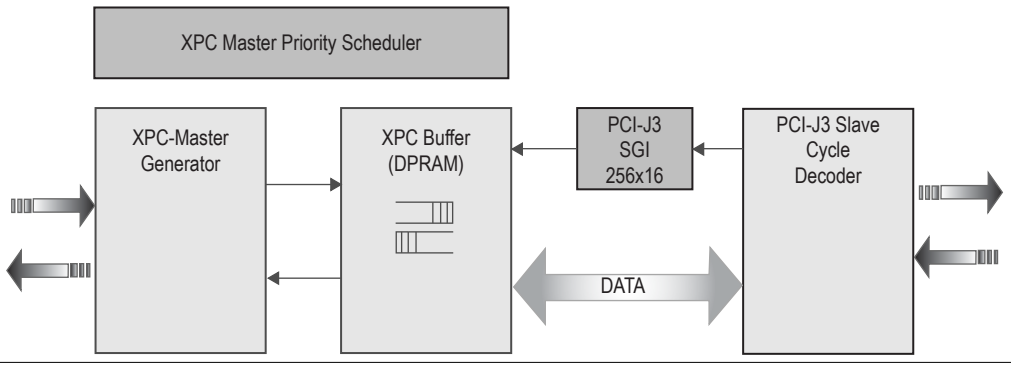
7.4.1 Functional Description

The PCI_J3 slave decoding uses:

- Base Address Register (BAR)
- Memory size (programmable from 32 to 1024 MBytes)

The PCI_J3-to-XPC write are completely decoupled through write-buffers (all writes are posted). The XPC-to-PCI_J3 read are isolated with a read-ahead buffer and can be programmed as read delayed (PCI Rev 2.2) or fully compelled (PCI 2.1). These two mechanisms allow to increase the general throughput of the PCI_J3 bridge. It also offers a better use of XPC-bus by providing full speed XPC-master interfacing.

Figure 7.2 PCI Slave



7.4.2 PCI_J3 XPC Master Cycle Scheduler

The XPC master scheduler is directly linked to the XPC arbitration request process. It evaluates continuously the local transactions pending following a pre-determined priority. In case of PCI_J3-to-XPC read pending and rflush active the scheduler first serve the xpcdirWBs to empty them before executing the xpcdirRD. The XPC arbitration implements four priority levels (4 down to 1) to minimize the latency.

Table 7-13 XPC Buffer

PRI_level	*Buffers*	*Description*	*XPC Arbitration*
4	xpcdirWB	PCI_J3-to-XPC read with flush pending. Serve first the xpcdirWB until no more available.	xpcREQ = 11
3	xpcdirRD	PCI_J3-to-XPC read pending	xpcREQ = 11
2	xpcdirWB	PCI_J3-to-XPC write direct buffer pending	xpcREQ = 10

7.4.3 PCI_J3 A32 Scatter / Gather IN

To map A32 PCI_J3 slave decoded window over the XPC bus, the missing informations are extracted from an in-coming scatter / gather. The in-coming scatter / gather is organized in 256x16, which provides 256 entries. The page size is conditioned by the PCI slave window size.

- 1 MByte, if the slave window is programmed to 32 to 256 MBytes.
- 4 MBytes, if the slave window is programmed to 512 to 1024 MBytes.

The XPC address translation is executed during the PCI_J3 slave access. The SGI initialization is handled through the internal registers PCI_J3_SGADD and PCI_J3_SGDAT (which are also accessible from the PCI_J3 CR/CSR configuration space). The SGI memory, which stores the 256 page entries, is built with fast dual-port SRAM.

Each page entry is associated with a direct mapped SGI page descriptor. The page descriptor is built over 16-bits, which provides support for the XPC cycle.

Table 7-14 PCI_J3 SGI Page Descriptor

Bits	*Name*	*Description*
[15:08]	xpcADD[31:24]	XPC top address remapping
[07:04]	xpcADD[23:20]	XPC top address remapping
[03:02]	TDST	XPC target destination = 00 : system memory (PowerPC bus) = 01 : local PCI bridge = 10 : extension PCI_J3 = 11 : do not use
[01]	xpcSNOOP	This bit is conditioned by xpcDST[1:0] If xpcDST[1:0] = 00 : enable snooping in PowerPCtarget If xpcDST[1:0] = 00 : (=0) PCI memory space (=1) PCI I/O space
[00]	xpcRFLUSH	This bit enable read with flush

NOTE

The SGI in XPC-PCI_J3 bridge is initialized with a remapping schema fullfilling the general XPC 32-bit mapping.

Since the SGI memory can also be initialized / modified from the PCI_J3 CONFIG space, it is possible to control the PCI-to-XPC remapping completely from the PCI_J3 side.

7.4.4 PCI_J3 Configuration Cycles

The PCI_J3 bridge implements a specific mechanism to support slave config space when it is plugged in the system controller slot (Slot 1). This mechanism allows access the system controller's configuration space from the peripheral slot, especially to the top area (0x80 -0xFF) holding the local registers.

Table 7-15 PCI _J3 Configuration

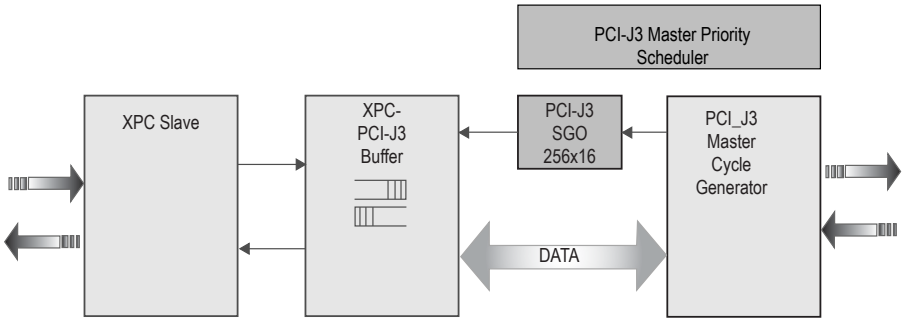
System Controller	*Description*
(PCI_J3/J3SYSEN = L)	Standard PCI selection with IDSEL selection (IDSEL extracted from pciAD[31:24])
(PCI_J3/J3SYSEN = H)	Selection with pciAD23 (virtual IDSEL extracted from pciAD23)

7.5 PCI_J3 Initiator Port (Master)

7.5.1 Functional Description

The PCI master interface handles the XPC-to-PCI transparent transactions and the embedded DMA transactions. For both transaction control, the PCI interface is isolated with an out-going scatter / gather (SGO) tailored in 16 MBytes page size.

Figure 7.3 PCI Master



The XPC-to-PCI_J3 write are completely decoupled through write-buffers (all writes are posted). The XPC-to-PCI_J3 read are isolated with a read-ahead buffer, these two mechanisms allow to increase the general throughput of the PCI_J3 bridge. It also provides a better use of XPC-bus by providing full speed XPC-slave interfacing.

7.5.2 PCI_J3 A32 Scatter / Gather OUT

The out-going scatter / gather is organized in 512x16, which provides 128 entries of 16 MBytes. The PCI_J3 address translation is executed during the PCI master access (and not during the XPC slave access).

The SGO initialization is handled through the internal registers PCI_J3_SGADD and PCI_J3_SGDAT (which are also accessible from the PCI CR/CSR space). The SGO memory, which stores the entries is built with fast dual-port SRAM, allowing fast translation.

Table 7-16 PCI_J3 SGO Page Descriptor

Bits	*Name*	*Description*
[63:40]	N/A	Not Used
[39:32]	pciADD[63:56]	PCI top address (reserved for PCI A64)
[31:16]	pciADD[47:32]	PCI top address (reserved for PCI A64)
[15:08]	pciADD[31:24]	PCI top address
[07:05]	A_Type	Address space field. = 000 : memory A32 = 001 : memory A64 = 010 : I/O = 011 : config = Others : reserved
[04:02]	D_Type	Data type encoding for PCI memory space = 000 : Std read / write = 001 : read multiple / write = 010 : read line/ write invalidate = Others : reserved
[01]	P_Type	PCI64 enable. If possible use PCI D64 transactions
[00]	N/A	Not Used

7.5.3 XPC-to-PCI_J3 Configuration Cycles

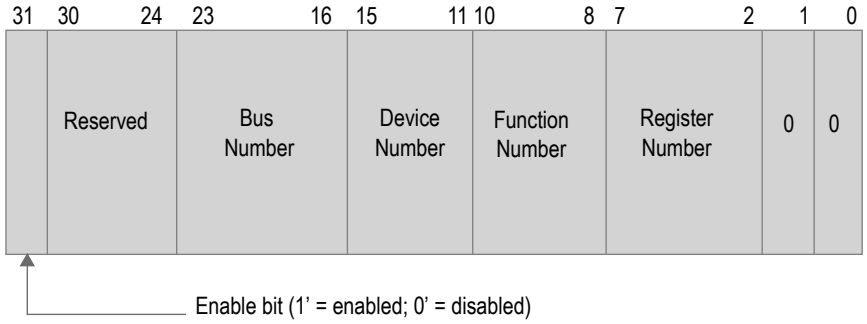
The PCI configuration cycles are generated through a 16 MBytes window programmed with AType = "011". Only the PCI_J3 system slot can do configuration cycles. The A23-00 address field is assigned as follows:

Table 7-17 PCI_J3 Configuration Cycle Encoding

Address bit	*Name*	*Description*
[31:24]	N/A	Not used in the 16 MBytes page
[23:16]	Bus Number	8-bit field selecting Type_0 or Type_1 configuration cycle type. If Bus_Number = 0x00, type_0 is issued with: Device_ID = 00000 : reserved 00001 : Slot 1 ! special 00010 : Slot 2 00011 : Slot 3 Others : not implemented
[15:11]	Device_ID	Device number
[10:9]	Function_ID	Function number
[8:2]	Register_ID	Register number
[1:0]		Fixed to “00”

NOTE The Type_0 cycle with Device_ID=00001 is used to access the configuration registers of the system controller.

Figure 7.4 PCI Configuration Cycle Encoding

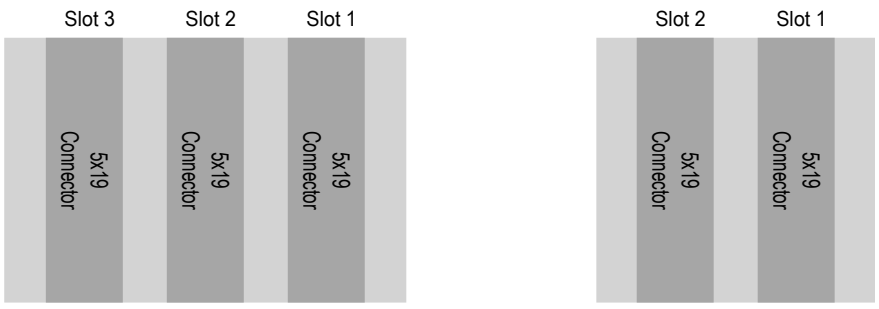


7.6 PCI_J3 Backplanes

The PCI_J3 backplane must be plugged in the rear of the CompactPCI backplane. These backplanes are similar to VSB backplane. Two PCI_J3 backplanes are available:

- BPA6413 Two slots PCI_J3
- BPA6414 Three slots PCI_J3

Figure 7.5 PCI_J3 Backplane



The PCI_J3 implements the PCI 2.2 protocol with limitation related to pciLOCK#.

7.6.1 PCI_J3 Connector Pin-Out

The above table supplies the pin-out designation. Row 'F' is completely grounded. Signals labelled in () are only available on Slot 1. For Slot 2: IDSEL = AD31. For Slot 3: IDSEL = AD30.

Table 7-18 PCI_J3 Pin-Out

pin	*A*	*B*	*C*	*D*	*E*
1	PRESENT#	serialIO	ENUM#		CLK
2	PERR#	SYSEN#	serialSYNC	(CLK_src)	PAR
3	CBE3	CBE2	CBE1	CBE0	AD31
4	AD30	AD29	AD28	AD27	AD26
5	AD25	AD24	AD23	AD22	AD21
6	AD20	AD19	AD18	AD17	AD16
7	AD15	AD14	AD13	AD12	AD11
8	AD10	AD9	AD8	AD7	AD6
9	AD5	AD4	AD3	AD2	AD1
10	AD0	TRDY#	IRDY#	AD63	AD62
11	AD61	AD60	AD59	AD58	AD57
12	AD56	AD55	AD54	AD53	AD52
13	AD51	AD50	AD49	AD48	AD47
14	AD46	AD45	AD44	AD43	ad42
15	AD41	AD40	AD39	AD38	AD37
16	AD36	AD35	AD34	AD33	AD32
17	PAR64	CBE7	CBE6	CBE5	CBE4
18	(GNT2#)/GNT#	(GNT3#)/SL23	RESET#	ACK64#	DEVSEL#
19	(REQ2#)/REQ#	(REQ3#)/IDSEL	REQ64#	STOP#	FRAME#

7.6.2 PCI_J3 Serial Bus

A defined serial bus interconnects all PCI_J3 boards. This serial bus is used to send / receive the four PCI INTx, the SERR# and a 'healthy' status. In normal operation, these status are sent back to the RIOC 4068 defined as system controller, but it is also possible to broadcast these information over the three PCI_J3 slots (all RIOC 4068 plugged on the PCI_J3 can receive the interrupts).
This serial bus is sequenced with the 33 MHz PCI_J3 clock and made up of two control signals, a synchronization signal (serialSYNC) and a data signal (serial_IO). The serialSYNC is driven by the system controller and serial_IO sequentially is driven by the 3 slots (1 to 3). Each slot issues sequentially their status information.

NOTE This serial bus mechanism introduces an additional latency of 500 µs during the dispatching of the PCI_J3 INTx#.

7.6.3 PCI_J3 Bridge Interrupts

The related PCI_J3 interrupts are all routed to the System Interrupt Controller as follows:

Table 7-19 PCI_J3 Interrupts

PCI_J3 Interrupts	*SIC ACTcode*	*Description*
PCIJ3_INTA	0x25	PCI_J3 INTA. Or function of all INTA coming from slot 1 + 2 + 3. Individual Interrupt pending status can be read in *PCI_J3_INTREQ*
PCIJ3_INTB	0x26	Idem for PCI_J3 INTB
PCIJ3_INTC	0x27	Idem for PCI_J3 INTC
PCIJ3_INTD	0x28	Idem for PCI_J3 INTD
PCIJ3_SERR	0x2C	Idem for PCI_J3 SERR

Table 7-19 PCI_J3 Interrupts

PCI_J3 Interrupts	*SIC ACTcode*	*Description*
PCIJ3_PERR	0x23	Not Used
PCIJ3_ENUM	0x58	Similar to CompactPCI ENUM#. Allows to detect PCI_J3 extraction
PCIJ3_MBIRQ	0x57	PCI_J3 mailbox interrupt

Please refer to the SIC Interrupt Controller, chapter 9.

NOTE 

On PEB 6418 used, as PMC carrier, the interrupt INTA - INTB - INTC - INTD are reassigned (rolled) on the top and on the bottom PMC to have different designation. Please refer to PEB User's Manual (6418/UM).

7.6.4 PPCMon Support for PCI_J3x

The onboard monitor PPCMon allows the configuration and the access of the PCI_J3 interface.

PPCMON "SET XPCI" COMMAND

The PPCMon command "set bridge_xpci" allows the configuration of the PCI_J3 interface at power-up. Any changes are executed immediately. Information kept by this command is stored in a non-volatile memory. The following subsets are used by the command:

```
ppcmon> set bridge_xpci <CR>
```

Table 7-20 Set XPCI Option

subset option	*Descriptions*
bridge_xpci_ena	Enable the PCI_J3 bridge interface. This option must be enabled only if a PCI_J3 backplane is installed. If disabled the XPC-to-CompactPCI bridge will occupy 2 GBytes **y**: PCI_J3 enabled **n**: PCI_J3 disabled
bridge_xpci_s64	Define the PCI_J3 data width **y**: PCI_J3 64-bit **n**: PCI_J3 32-bit
bridge_xpci_size	Define the PCI target window size. Defined as: **32**, **64**, **128**, **256**, **512** : size in MByte
bridge_xpci_prefetch	Set block size of PCI target read prefetch. Defined as: **12**, **32**, **64**, **128** : Bytes if D32 **32**, **64**, **128**, **256** : Bytes if D64
bridge_xpci_flush	Define the XPC-PCI_J3 flush policy. Defined as: **y**: read flush only over direct PowerPC to PCI_J3 writes **n**: read flush over PowerPC and DMA writes
bridge_xpci_mretry	Not implemented in XPC-to-XPCI bridge
bridge_xpci_sretry	Not implemented in XPC-to-XPCI bridge
bridge_xpci_timer	Define the disconnect timer value. If the read are defined with "bridge_xpci_rcmpld = y", the short value should be selected **2E10**: short value **2E16**: long value, PCI 2.2 with uncompelled read
bridge_xpci_parity	Define the XPC - PCI_J3 parity reporting policy **y**: parity error over PCI is reported to XPC **n**: parity error over PCI is not reported over XPC
bridge_xpci_rcmpld	Define the READ policy **y**: read compelled (PCI 2.1) **n**: read uncompelled (PCI 2.2)

NOTE 

Many of above options are directly related to bits located in the PCI_J3_LOC_CTL register.

PPCMON ACCESS COMMANDS

The PPCMon command px, dx, tx ,... allows the user to directly access the PCI_J3 without requiring any SGO initialization. All of the commands have the same synthax, as follows:

```
ppcmon>px.s addr data mode <CR>
```

with the following signification:

s	b, h, l	Size as byte, halfword, long
addr		Address in hexadecimal value (4,6 or 8 digit)
data		Data in hexadecimal value (2, 4 or 8 digit)
mode		Mode in alphanumeric as mem or io

Table 7-21 Basic PCI_J3 Access Debugging Command

Command	*Description*
px	Patch PCI_J3. Dynamic read / write access over PCI_J3 px.l 82030000 ? mem <CR> ; PCI_J3 read long word px.b 82030000 12 mem <CR> ; PCI_J3 memory write
dx	Display PCI_J3. Display value of PCI_J3 location dx.l 84030000 <CR> ; Display 80 consecutive location dx.l 84030000..84031000 <CR> ; Display specified range

PPCMon uses only the first PCI_J3 page 0x8000'0000-0x80FF'FFFF. The scatter / gather page descriptor is reloaded continuously.

NOTE For complete PPCMon command list, please refer to the PPCMon User's Manual (Ref. DOC 3317C/UI).

PPCMON STATUS & CONFIG COMMANDS

The PPCMon command "xpci config" displays the current setting of the PCI_J3 interface. It can provide a quick debugging status of the onboard XPC-to-PCI_J3 bridge.

```
ppcmon>xpci config <CR>

XPCI interface [enabled]
System Controller
Slave      : A32 Size = 32 Mbytes
             A32 Base = 0x80000000
             A64 Base = 0x00000000'00000000  [disabled]
```

The PPCMon command `xpci show` is prints out the PCI_J3 physical allocation.

```
ppcmon>xpci show <CR>
```

The PPCMon command `diag xpci` also scans the PCI_J3 backplane, executes the PCI configuration (physical memory allocation) and prints out foreseen allocations.

Chapter 8
Local PCI (PCI#1)

8.1 Introduction

The Local_PCI bus (64-bit 33 MHz) is driven by a XPC-to-Local_PCI bridge. This Local_PCI bus connects:
- Local SubI/O interface
- Two IEEE P1386 PMC sockets
- PCI Ethernet controller AMD79C973

To achieve maximum performance the Local_PCI and the XPC bus are isolated / buffered through synchronous dual-port SRAM. This memory device is organized in multiple buffers used as 'write-buffer' or 'read-ahead buffer'. This architecture allows to build an interface, which runs at its maximum time-domain frequency.

The Local_PCI bridge also has its own PCI configuration registers, in such a way that these registers can be accessed from an intelligent PMCs. Since this bridge must be initialized at power-up to provide access to the boot PROM and I/O resources, default values are stored in the PCI configuration registers.

The PCI configuration registers follows the PCI 2.2 specification.

8.1.1 Local_PCI Internal Registers

All internal registers are dual ported between the PCI config and the XCSR. This allows for 'intelligent PMCs, such as MFCC, to access directly the XPC_PCI bridge's configuration registers.

Table 8-1 Local_PCI Internal Registers

Local_PCI Registers	*Local XCSR mapping (PowerPC Address)*	*Local_PCI Configuration mapping (offset)*
PCI CONF Registers	0xF900'9000 to 0xF900'907F	0x00 to 0x7F
Local_PCI_LOC_CTL	0xF900'9080	0x80
Local_PCI_LOC_STA	0xF900'9084	0x84
Local_PCI_SGADD	0xF900'9088	0x88
Local_PCI_SGDAT	0xF900'908C	0x8C

NOTE

The access to these registers requires swapping in order to match the PCI / PowerPC endian conversion. PPCMon command should be used as follows:

```
ppcmon>pm.ls f9009000 <CR>
0xf9009000 : 406510d9 ->;Result as defined by PCI format
```

8.1.2 Local_PCI Local Registers

LOCAL_PCI_LOC_CTL

Table 8-2 Major XPC-PCI Bridge's Control Bits (read / write register)

	LOC_CTL CPU = 0xF900'9080 & Local_PCI = CONF_Base + 0x80		
Bits	***Name***	***Mode***	***Description***
[31:23]	N/A	-	Not Used
[22]	extPMC2ndENA	R/W	PMC second device arbitration enable. The new PPMC spec allows two PCI devices on the PMC = 0 : secondary device not enabled = 1 : secondary device enabled
[21]	ctlpciCOMPREAD	R/W	Local_PCI slave read mode of operation = 0 : standard 2.2 delayed read = 1 : read compelled (full handshake)
[20]	xpcGEN_PARI	R/W	XPC parity generation = 0 : XPC parity always generated OK = 1 : XPC parity generated OK if related PCI cycle have PCI parity OK, otherwise XPC parity are inverted.
[19:16]	N/A	-	Not Used
[15]	ctlpciDISCTIM	R/W	Local_PCI read slave discard timer. As defined by the PCI specification 2.2 = 0 : 2E16 clocks = 1 : 2E10 clock (short)
[14]	N/A	-	Not Used
[13]	N/A		Not Used
[12]	pciLOCAL_RSTcmd	R/W	When written to '1', a 120 us PCI reset pulse is issued. The "pciLOCAL_RSTcmd" control bit remains at 1 during the PCI reset assertion.
[11:10]	pciARB_MOD	R/W	Internal Local_PCI central arbiter option = 00 : standard pipelined + bus parking on XPC-LocalPCI bridge. = 01 : not pipelined + bus parking on dummy devices assuring the pull-up of active signals. = 10 : reserved for debugging = 11 : pipelined + bus parking on latest LocalPCI master (PCI 2.2 spec)
[09:08]	xpcARBMOD	R/W	XPC bus central arbitration mode of operation: = 00 : standard pipelined = 01 : fairness = 1x : reserved
[07]	LocPCI_FLUSH_MOD	R/W	Internal read / write priority resolution for XPC to Local_PCI master transactions. = 0 : read flushing only over pcidirWB (CPU write transactions) = 1 : read flushing over pcidirWB and pcidmaWB (CPU and DMA write transactions)
[06:05]	READ_PFETCH_SIZ	R/W	Local_PCI slave read Prefetch size (In 32-bit word) = 00 : 6 if PCI32, 8 if PCI64 = 01: 8 if PCI32, 16 if PCI64 = 10 : 16 if PCI32, 32 if PCI64 = 11 : 32 if PCI32, 64 if PCI64
[04:02]	N/A	-	Not Used
[01]	N/A	-	Not Used
[00]	pciSIZ64_ENA	R/W	Local_PCI backplane size 64-bit enabled. This control bit is set after power-up reset. Can be reset in special situation.

NOTE Default value for Local_PCI_LOC_CTL = 0x0010'4000

LOCAL_PCI_LOC_STA

Table 8-3 XPC to Local_PCI Bridge Basic Status Information (read only register)

	$LOC\_STA_{CPU = 0xF900'9104 \quad \& \quad Local\_PCI0 = CONF\_Base + 0x84}$		
Bits	***Name***	***Mode***	***Description***
[31:12]	N/A	-	Not Used
[11]	Local_PCI_ARB_TO	R/Wc	Local_PCI arbitration time-out flag. Set to 1 if a pciGNT# is issued and no PCI cycle is started after 16 clock cycle. This flag is cleared by writing 1 over it.
[10]	Local_PCI_READTO	R/Wc	Local_PCI discard timer time-out flag. This flag is cleared by writing 1 over it.
[09]	XPX_WRERR	R/Wc	Local_PCI to XPC write error detected. This flag is set in case of XPC error. This flag is cleared by writing 1 over it.
[08]	Local_PCI_WRERR	R/Wc	XPC to Local_PCI write error detected. This flag is set in case of Local_PCI error (target abort or master abort).This flag is clear by writing 1 over it.
[07:00]	N/A	-	Not Used

LOCAL_PCI_SGADD

This register holds the local address pointer for internal SRAM, which holds the page descriptors associated with the scatter / gather in and out.

Table 8-4 Local Address Pointer for Internal SRAM

	$SGADD_{CPU = 0xF900'9088 \quad \& \quad Local\_PCI = CONF\_Base + 0x88}$		
Bits	***Name***	***Mode***	***Description***
[31:26]	N/A	-	Not Used
[25]	localPMC_BMOD1_1	-	From PMC Slot#1
[24]	localPMC_BMOD1_0	-	From PMC Slot#0
[23:18]	N/A	-	Not Used
[17:16]	InitRAM_SEL	R/W	This 2-bit field select the internal SRAM to initialize = 00 : SGO (XPC to PCI scatter / gather) = 01 : SGI (PCI to XPC scatter / gather) = 1x : Reserved
[15:09]	N/A	-	Reserved for future extension
[08:00]	sgREG_ADD	R/W	Address pointer to SRAM holds the page descriptor.

LOCAL_PCI_SGDAT

This 16-bit register allows to read / write the internal SRAM SGO and SGI for initialization. The SRAM location is pointed by the previous register SGADD.

Table 8-5 Internal SRAM SGO and SGI Initialization (16-bit register read / write)

	$SGDAT_{CPU = 0xF900'908C \quad \& \quad Local\_PCI = CONF\_Base + 0x8C}$		
Bits	***Name***	***Mode***	***Description***
[31:16]	N/A	-	Not Used
[15:00]	sgREG_DATA	R/W	Data register for local SRAM access (read / write)

8.1.3 Local_PCI Initiator and Scatter / Gather OUT

The XPC to Local_PCI transaction incorporates a scatter / gather, which can remap the XPC address to PCI address. The scatter / gather provides 64 pages of 16 MBytes over the XPC-to-Local_PCI address space (0xC0000'0000 - 0xFFFF'FFFF).

By default, the scatter / gather memory is initialized with the following address translation and XPC destination agent:

Table 8-6 Local PCI Address Mapping

XPC Address	*PCI Destination & PCI Physical Address*
0x0000'0000 - 0xBFFF'FFFF	Not decoded
0xC000'0000 - 0xF5FF'FFFF	PCI Memory: 0xC000'0000 - 0xF5FF'FFFF
0xF600'0000 - 0xF6FF'FFFF	PCI IO: 0xF600'0000 - 0xF6FF'FFFF
0xF700'0000 - 0xF7FF'FFFF	PCI Config: 0x8000'0000 - 0x80FF'FFFF
0xF800'0000 - 0xF8FF'FFFF	PCI Memory: 0xF800'0000 - 0xF8FF'FFFF
0xF900'0000 - 0xF9FF'FFFF	PCI IO: 0xF900'0000 - 0xF9FF'FFFF
0xFA00'0000 - 0xFFFF'FFFF	PCI Memory: 0xFA00'0000 - 0xFFFF'FFFF

NOTE The fields P_Type and D_Type are fixed to 0.

The page descriptor is built on 64 bits with the following bit allocation:

Table 8-7 Local_PCI SGO Page Descriptor

Bits	*Name*	*Description*
[63:16]	Reserved	Not used in XPC-to-Local_PCI bridge.
[15:08]	pciADD[31:24]	PCI top address
[07:05]	A_Type	Address space field. = 000 : memory A32 = 001 : reserved (memory A64) = 010 : I/O = 011 : config = others : reserved
[04:02]	D_Type	Data type encoding for PCI memory space = 000 : Std read / write = 001 : read multiple / write = 010 : read line / write invalidate = others : reserved
[01]	P_Type	PCI64 enable. If possible, use of PCI D64 transactions
[00]	N/A	Not Used

NOTE The scatter / gather memory is implemented with same bit mapping as in PCI_J3 and CompactPCI.

XPC-TO-LOCAL_PCI CONFIGURATION CYCLES

The PCI configuration cycles are generated through the 16 MBytes window programmed with AType = "011" (0xF700'0000-0xF7FF'FFFF). The A23-00 address field is assigned as follows:

Figure 8.1 PCI Configuration Cycles

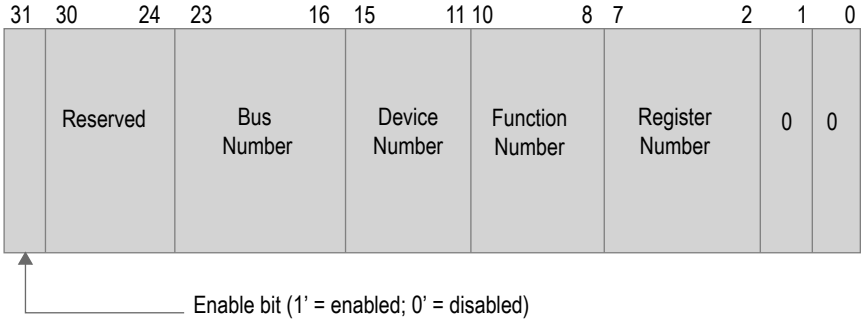


Table 8-8 LocalPCI Configuration Cycle Encoding

Address bit	*Name*	*Description*
[23:16]	Bus Number	8-bit field selecting Type_0 or Type_1 configuration cycle type. If Bus_Number = 0x00, Type_0 is issued with: Device_ID = 00000 : reserved 00001 : Ethernet 79C973 00010 : PMC_1 00011 : PMC_2 00100 : PMC_1b (second device PPMC) 00101 : PMC_2b (second device PPMC)
[15:11]	Device_ID	Device number
[10:09]	Function_ID	Function number
[08:02]	Register_ID	Register number
[01:00]		Fixed to 00

NOTE Refer to PCI Specification for complete explanation about Type_0 & Type_1.

8.1.4 Local_PCI Target and Scatter / Gather IN

The PCI target interface bridge PCI memory address 0x0000'0000 to 0xBFFF'FFFF back to the XPC (only the PCI memory space is decoded). The PCI-to-XPC bridge incorporates a SGI, which can remap the PCI address to XPC address. By default, the scatter / gather memory is initialized with the following address translation and XPC destination agent:

Table 8-9 Scatter / Gather Addresses

PCI Mem Address	*XPC Destination*
0x0000'0000 - 0x3FFF'FFFF	System memory: 0x0000'0000 - 0x3FFF'FFFF
0x4000'0000 - 0x7FFF'FFFF	CompactPCI: 0x4000'0000 - 0x7FFF'FFFF
0x8000'0000 - 0xBFFF'FFFF	PCI_J3: 0x8000'0000 - 0xBFFF'FFFF
0xC000'0000 - 0xFFFF'FFFF	Not decoded.

The full PCI target address space is divided in 192 pages of 16 MBytes (the top 64 pages is reserved for local addressing). Each 16 Mbytes page entry is associated with a direct mapped SGI page descriptor. The page descriptor is built over 16 bits providing support for the XPC cycle built.

Table 8-10 Local_PCI SGI Page Descriptor

Bits	*Name*	*Description*
[15:04]	xpcADD[31:24]	XPC top address remapping
[07:04]	not used	Reserved
[03:02]	TDST	XPC target destination = 00 : System memory (PowerPC bus) = 01 : Do not use = 10 : Extension PCI_J3 = 11 : CompactPCI
[01]	xpcSNOOP	This bit is conditioned by xpcDST[1:0] If xpcDST[1:0] = 00: enable snooping in PowerPC target If xpcDST[1:0] = 00: (=0) PCI memory space (=1) PCI IO space
[00]	xpcRFLUSH	This bit enable read with flush

8.2 Ethernet Controller

8.2.1 Ethernet

The RIOC 4068 board hosts the AMD79C973 ethernet controller providing access to 10/100 Base-Tx networks through a front panel RJ45 connector.

FEATURES

- PCI 2.1 compliant 32-bit bus interface at 33 MHz
- High-performance PCI bus master with integrated DMA
- Ethernet 10 Base-T and 100 Base-Tx auto-negotiation

Interface Connector RJ45, 8-pin connector

Data Bit Rate 10 Base-T at 10 Mbits/s
100 Base-Tx at 100 Mbits/s
IEEE 802.3 auto-negotiation

External Cables Category 3 unshielded twisted pair cable
1000 meter maximum for 10 Base-T operation
Category 5 unshielded twisted pair cable
100 meter maximum for 100 Base-Tx operation

The ethernet device is fully initialized at power on or after a reset, by the onboard PPCMon firmware.

Network parameters (such as IP address) can be set using PPCMon commands (`set inet`, etc.). Please refer to the PPCMon documentation for more information (Ref. DOC 3317C/UI).

This allows for booting the OS from the network, downloading user-code into the Flash EPROM of the board, etc.

CES Board Support Packages (BSP) for VxWorks, LynxOS and Linux contain drivers for the AMD79C973, which are automatically started during the boot of the OS, in order to provide direct network access.

Users who would like to write their own drivers should download the AMD79C973 data sheet from the AMD web site to get a full documentation for programming the chip.

8.3 PMCs Slot

8.3.1 PCI 64-bit Slot with PPMC Extension

Both of the local PMC slots are compliant with the PCI specification for 64 bits local PCI at 33 MHz using 3.3 Volts signaling, in addition the top PMC provides + 5 Volts tolerant interface signals. Provision has been made on both PMC slots to support the PPMC standard.

Table 8-11 PPMC Additional Signals

PMC Connector	*Signal Name1*
Jn2 / pin_52	REQ2 #
Jn2 / pin_54	GNT2 #
Jn2 / pin_58	EREADY
Jn2 / pin_60	RST_REQ #
Jn2 / pin_64	MONARCH #

Please refer to the IEEE P1386 PPMC Specifications for more information.

8.3.2 PMC Interrupt Routing

The interrupts controller provides 5 interrupt request lines for the two onboard PMCs. The interrupts allocations ease the software recognition even for PMC's with more than one interrupt request.

Table 8-12 PMC Interrupt Allocation

SIC Interrupt	*PMC #1*	*PMC #2*
Local_PMC1_INTA	PCI_INTA	---
Local_PMC2_INTA	---	PCI_INTA
Local_PCI_INTB	PCI_INTB	PCI_INTD
Local_PCI_INTC	PCI_INTC	PCI_INTB
Local_PCI_INTD	PCI_INTD	PCI_INTC
Local_PCI_SERR	PCI_SERR	PCI_SERR

NOTE The other local PCI agents do not share these interrupt lines.

8.3.3 Power Supply 3.3V and 5V

The +3.3 Volts is connected directly to the CompactPCI power supply through the Live Insertion Controller. The maximum current is limited by the PMC Spec at 2.3 Amps for each PMC slot.

The +5 Volts is connected directly to the CompactPCI power supply through the Live Insertion Controller. The maximum current is limited by the PMC Spec at 1.5 Amps for each PMC slot.

NOTE The PMC $V_{I/O}$ power supply is connected to the PMC 3.3 Volts DC / DC.

Table 8-13 PMC Spec. Power Limitation

Power Supply	*Voltage (Typical)*	*Current (Max)*
PMC#1 & PMC#2	3.3 Volts	2.3 Amps (1)
PMC#1 & PMC#2	+ 5 Volts	1.5 Amps

(1) Note: Maximum current is for each PMC.

Chapter 9
Interrupt Structure

9.1 Interrupt Controller

The interrupt controller of the RIOC 4068 is an enhanced version of the SIC 6351, which equipped the previous generation of CES RIO2 processor boards.

Compatibility is maintained but new features have been added.

FEATURES

- Controls up to sixty-three interrupt sources on seven interrupt levels
- Supports 16-bit interrupt vectors for programmed IACK cycles
- Provides 16-bit vectors for each sixty-three interrupt sources and one spurious interrupt
- Supports auto-masking on the interrupt levels 7 to 1
- Provides auto-disabling interrupt logic during IACK cycles
- Provides 32-bit global-masks to enable / disable all of the sixty-three interrupts
- Provides 32-bit global-status to monitor all of the sixty-three interrupts
- Provides seven timer interrupts (3 x auto-reloaded IRQs and 3 x auto-cleared IRQs and 1 x TICK Timer)

NEW REGISTERS

- 32-bit global-mask registers (GBL_MSK0, GBL_MSK1) to enable / disable interrupts
- 32-bit global-status registers (GBL_STA0, GBL_STA1) to monitor interrupts pending
- 8-bit auto-mask register
- 8-bit SMI register (CPU SMI request)
- 8-bit Timer registers
- 8-bit LED control register (available on certain part numbers only)

NEW FEATURES

- Automatic interrupt level masking (same as the RIO2)
- Automatic clear of the edge-sensitive interrupt after IACK cycles
- Automatic disable of the ICR enable bit for the interrupt after IACK cycles
- Expansion of the interrupt sources and vectors for a maximum of sixty-three interrupts
- Interrupt level 7 only request an interrupt on the CPU MCP signal (NMI)
- Interrupt levels 6 to 1 request interrupts on the CPU INT0 signal (INT)
- 3 x auto-cleared interrupt timers from 10 µs up to 100 ms
- 3 x auto-reloaded interrupt timers from 10 µs up to 100 ms
- 1 x TICK Timer from 10 µs up to 100 ms

9.2 Mode of Operation

The SIC has been designed by CES to act as a complete interrupt controller. It can control up to sixty-three interrupt sources and can dispatch them to the two CPU interrupt lines (MCP, INT) through programmable register options. Each interrupt source can be set to work with an internal vector (vectors supplied by the internal SIC's SRAM).

Each interrupt source is associated with a Interrupt Cell Register (also know as ICRx), containing information such as input polarity, edge or level sensitivity, destination, destination relative priority (interrupt level), masked or enabled. The ICRx controls the conditions under which an interrupt is routed to the CPU.

Interrupt levels 1 to 6 generate INT interrupts on the PowerPC, whereas level 7 is reserved to generate MCP interrupts to the CPU. SMI interrupts can be generated by using the SIC SMI register.

When interrupted, the CPU acknowledges the interrupt by reading the interrupt acknowledge register (IACK-REG) of the SIC, which returns the vector of the highest priority interrupt currently pending. The interrupt routine should then clear the interrupt source, take the appropriate action, and then re-enable the interrupt.

9.2.1 Interrupt Scanning

The SIC Interrupt Controller logic detects the pending interrupts by scanning continuously the interrupt levels from 7 down to 1. When an interrupt request has been detected, the scan-logic stops until the interrupt is acknowledged (Iack) or cleared.

WARNING

The scan-logic will latch the interrupt request when detected (and not masked), even for the level-sensitive, which shall not be cleared before the IACK cycle.

9.2.2 Automatic Clear / Disable

The Interrupt Acknowledge cycle (IACK) performs the following functions:

- The edge sensitive interrupt is automatically cleared
- The corresponding Interrupt Cell Register (ICR) is automatically disabled

WARNING

The interrupt routine will first clear the level-sensitive interrupt source, then it will re-enable the corresponding ICRx.

9.2.3 Automatic Mask

The mode “Auto-Mask” allows the software to automatically manage the masking-up and the masking-down of the incoming interrupt requests.

NOTE

The interrupt under process will rise-up to its level of the interrupt level-mask after the IACK cycle. Only the higher-level interrupt requests are considered, until the interrupt routine sets the level-mask to a lower value.

9.2.4 Interrupt Polling

The global status registers STAT_REG0 and STAT_REG1 allow the software applications to poll the state of the all sixty-three interrupt requests connected to the SIC. The status of the interrupt request lines is never masked (even if disabled). The polarity defined in the corresponding ICRx is considered to set to “one” the corresponding bits into the status registers.

NOTE

The bit read as “one” always means that the Interrupt request is (still) asserted, if the polarity bit in the corresponding ICRx is set correctly.
Beware: In this mode the Edge Sensitive Interrupts shall be cleared by using Clear Register (CLR0 & CLR1)

9.3 Internal Registers

9.3.1 SIC Register Mapping

Table 9-1 SIC Register Mapping

CPU	*Register Name*
0xF900'0000 - 0xF900'00FC	Vector Assignment
0xF900'0100 - 0xF900'0108	Interrupt Global Control and Status
0xF900'0200	Auto-Mask Level Function Control
0xF900'0300	CPU SMI Interrupt Control
0xF900'0400 - 0xF900'04FC	ICRx: Individual Interrupt Cell
0xF900'0500	Thermo IRQ Register
0xF900'0600	IACK
0xF900'0700	Revision Register
0xF900'0800 - 0xF900'080C	Global Interrupt Mask and Status

9.3.2 Interrupt Source to Vector Assignment

The following table contains null data after reset. Both PPCMon and the OS running on the RIOC 4068 initialize this table to default values, which the user should not modify.

Table 9-2 SIC Vector Assignment

Offset	*Bit Wide*	*Interrupt Source*	*ACTCODE*
0x0000	D16	Spurious IT Vector	0x00
0x0004	D16	FIFO#0_EMPTY	0x01
0x0008	D16	FIFO#1_EMPTY	0x02
0x000C	D16	FIFO#2_EMPTY	0x03
0x0010	D16	FIFO#3_EMPTY	0x04
0x0014	D16	FIFO#4_EMPTY	0x05
0x0018	D16	FIFO#5_EMPTY	0x06
0x001C	D16	FIFO#6_EMPTY	0x07
0x0020	D16	FIFO#7_EMPTY	0x08
0x0024	D16	FIFO#0_FULL	0x09
0x0028	D16	FIFO#1_FULL	0x0A
0x002C	D16	FIFO#2_FULL	0x0B
0x0030	D16	FIFO#3_FULL	0x0C
0x0034	D16	FIFO#4_FULL	0x0D
0x0038	D16	FIFO#5_FULL	0x0E
0x003C	D16	FIFO#6_FULL	0x0F
0x0040	D16	FIFO#7_FULL	0x10
0x0044	D16	LAN_CTRL (ethernet)	0x11
0x0048	D16	SPARE (Pullup 4K7)	0x12
0x004C	D16	SIO#1 (console)	0x13
0x0050	D16	SIO#2 (auxiliary)	0x14
0x0054	D16	PMC#1_INTA	0x15
0x0058	D16	PMC#2_INTA	0x16
0x005C	D16	PMC#1_INTB / PMC#2_INTD	0x17
0x0060	D16	PMC#1_INTC // PMC#2_INTB	0x18
0x0064	D16	PMC#1_INTD // PMC#2_INTC	0x19
0x0068	D16	CPCI_ENUM#	0x1A

Table 9-2 SIC Vector Assignment

Offset	*Bit Wide*	*Interrupt Source*	*ACTCODE*
0x006C	D16	CPCI_SERR2	0x1B
0x0070	D16	CPCI_PERR#	0x1C
0x0074	D16	CPCI_INTD#	0x1D
0x0078	D16	CPCI_INTC#	0x1E
0x007C	D16	CPCI_INTB#	0x1F
0x0080	D16	CPCI_INTA#	0x20
0x0084	D16	CPCI_BMA	0x21
0x0088	D16	LOCAL_PCI_PERR	0x22
0x008C	D16	P0_PCI_PERR	0x23
0x0090	D16	THERMO_IRQ	0x24
0x0094	D16	J3_PCI_INTA	0x25
0x0098	D16	J3_PCI_INTB	0x26
0x009C	D16	J3_PCI_INTC	0x27
0x00A0	D16	J3_PCI_INTD	0x28
0x00A4	D16	TICK_TIMER_IRQ	0x29
0x00A8	D16	RTC_NVRAM_IRQ	0x2A
0x00AC	D16	LOCAL_PCI_SERR	0x2B
0x00B0	D16	J3_PCI_SERR	0x2C
0x00B4	D16	CPCI_SPARE	0x2D
0x00B8	D16	CPCI_DEG#	0x2E
0x00BC	D16	CPCI_FAL#	0x2F
0x00C0	D16	TIME_INT#0	0x30
0x00C4	D16	TIME_INT#1	0x31
0x00C8	D16	TIME_INT#2	0x32
0x00CC	D16	TIME_INT#3	0x33
0x00D0	D16	TIME_INT#4	0x34
0x00D4	D16	TIME_INT#5	0x35
0x00D8	D16	J3_PCI_SPARE	0x36
0x00DC	D16	XPC_CPU_DMA_IRQ	0x37
0x00E0	D16	RS485_IRQ	0x38
0x00E4	D16	J3_PCI_MAILBOX	0x39
0x00E8	D16	J3_PCI_ENUM	0x3A
0x00F0 - 0x00FC	D16	Not Used	0x3B - 0x3F

9.3.3 Interrupt Cell Registers

Each interrupt source can be configured independently.

Table 9-3 Generic ICR Register

Bits	*Name*	*Mode*	*Description*
[07]	ENABLE	R/W	Source gate: 0 = interrupt masked 1 = interrupt enabled
[06]	Not Used	R	Read as 0
[05]	EDGE	R/W	Edge / level selection: 0 = level-sensitive 1 = edge-sensitive

Table 9-3 Generic ICR Register

Bits	*Name*	*Mode*	*Description*
[04]	POLARITY	R/W	Polarity selection: 0 = active low 1 = active high
[03]	Not Used	R	Read as 0
[02:00]	level [02:00]	R/W	Destination level. This 3-bit field encodes the relative priority of the considered cell: 000 = disabled 001 = lower priority ... 111 = highest priority

NOTE

The edge bit can be combined with the polarity:

Edge	Polarity	
0	0	Level active low
0	1	Level active high
1	0	Negative edge
1	1	Positive edge

The ICR's enable control-bit allows to enable / mask the interrupt sources. Enabling a source means that if the input cell is activated, it will generate an interrupt request to the CPU with the associated priority specified by ICR level [02:00]. If the source is masked, it will not generate the interrupt, although the status of the cell will still be reflected in the corresponding SSRx register.

Table 9-4 ICRs Address Mapping

Offset	*ICR*	*Bit Wide*	*Interrupt Source*	*ACTCODE*
0x0400	ICR_1	D08	FIFO#0_EMPTY	0x01
0x0404	ICR_2	D08	FIFO#1_EMPTY	0x02
0x0408	ICR_3	D08	FIFO#2_EMPTY	0x03
0x040C	ICR_4	D08	FIFO#3_EMPTY	0x04
0x0410	ICR_5	D08	FIFO#4_EMPTY	0x05
0x0414	ICR_6	D08	FIFO# _EMPTY	0x06
0x0418	ICR_7	D08	FIFO#6_EMPTY	0x07
0x041C	ICR_8	D08	FIFO#7_EMPTY	0x08
0x0420	ICR_9	D08	FIFO#0_FULL	0x09
0x0424	ICR_10	D08	FIFO#1_FULL	0x0A
0x0428	ICR_11	D08	FIFO#2_FULL	0x0B
0x042C	ICR_12	D08	FIFO#3_FULL	0x0C
0x0430	ICR_13	D08	FIFO#4_FULL	0x0D
0x0434	ICR_14	D08	FIFO#5_FULL	0x0E
0x0438	ICR_15	D08	FIFO#6_FULL	0x0F
0x043C	ICR_16	D08	FIFO#7_FULL	0x10
0x0440	ICR_17	D08	LAN_IRQ (Ethernet)	0x11
0x0444	ICR_18	D08	SPARE (Pull-up 4K7)	0x12
0x0448	ICR_19	D08	SIO#1 (Console)	0x13
0x044C	ICR_20	D08	SIO#2 (Auxiliary)	0x14
0x0450	ICR_21	D08	PMC#1_INTA	0x15
0x0454	ICR_22	D08	PMC#2_INTA	0x16
0x0458	ICR_23	D08	PMC#1_INTB / PMC#2_INTD	0x17
0x045C	ICR_24	D08	PMC#1_INTC // PMC#2_INTB	0x18
0x0460	ICR_25	D08	PMC#1_INTD // PMC#2_INTC	0x19
0x0464	ICR_26	D08	CPCI_ENUM#	0x1A

Table 9-4 ICRs Address Mapping

Offset	*ICR*	*Bit Wide*	*Interrupt Source*	*ACTCODE*
0x0468	ICR_27	D08	CPCI_SERR2	0x1B
0x046C	ICR_28	D08	CPCI_PERR#	0x1C
0x0470	ICR_29	D08	CPCI_INTD#	0x1D
0x0474	ICR_30	D08	CPCI_INTC#	0x1E
0x0478	ICR_31	D08	CPCI_INTB#	0x1F
0x047C	ICR_32	D08	CPCI_INTA#	0x20
0x0480	ICR_33	D08	CPCI_BMA	0x21
0x0484	ICR_34	D08	LOCAL_PCI_PERR	0x22
0x0488	ICR_35	D08	J3_PCI_PERR	0x23
0x048C	ICR_36	D08	THERMO_IRQ	0x24
0x0490	ICR_37	D08	J3_PCI_INTA	0x25
0x0494	ICR_38	D08	J3_PCI_INTB	0x26
0x0498	ICR_39	D08	J3_PCI_INTC	0x27
0x049C	ICR_40	D08	J3_PCI_INTD	0x28
0x04A0	ICR_41	D08	TICK_TIMER_IRQ	0x29
0x04A4	ICR_42	D08	RTC_NVRAM_RQ	0x2A
0x04A8	ICR_43	D08	LOCAL_PCI_SERR	0x2B
0x04AC	ICR_44	D08	J3_PCI_SERR	0x2C
0x04B0	ICR_45	D08	CPCI_SPARE	0x2D
0x04B4	ICR_46	D08	CPCI_DEG#	0x2E
0x04B8	ICR_47	D08	CPCI_FAL#	0x2F
0x04BC	ICR_48	D08	TIME_INT#0	0x30
0x04C0	ICR_49	D08	TIME_INT#1	0x31
0x04C4	ICR_50	D08	TIME_INT#2	0x32
0x04C8	ICR_51	D08	TIME_INT#3	0x33
0x04CC	ICR_52	D08	TIME_INT#4	0x34
0x04D0	ICR_53	D08	TIME_INT#5	0x35
0x04D4	ICR_54	D08	J3_PCI_SPARE	0x36
0x04D8	ICR_55	D08	XPC_DMA_IRQ	0x37
0x04DC	ICR_56	D08	RS485_IRQ	0x38
0x04E0	ICR_57	D08	J3_PCI_MAILBOX#	0x39
0x04E4	ICR_58	D08	J3_PCI_ENUM#	0x3A
0x04E8- 0x04F8	ICR_59 to ICR_63	D08	Not Used (Reserved)	0x3B - 0x3F

The interrupt sources of the RIOC 4068 are tailored into eight groups, which correspond to a given initialization of the ICR registers.

Table 9-5 Interrupt Sources Groups

Cell Type	*Interrupt Request*
ICR Type C1	CPCI_FAL#
ICR Type D	CPCI_DEG#
ICR Type E	PCI_PERR#, PCI_SERR#
ICR Type EP	TIME_INT bits [05:00], TICK_TIMER
ICR Type F	CPCI_INTA, B, C, D RTC, LAN PCI_INTA, INTB, INTC, INTD, All of the other standard IRQs

Table 9-5 Interrupt Sources Groups

Cell Type	*Interrupt Request*
ICR Type FE	SIO#1, SIO#2
ICR Type G	RS485_IRQ / BLK_BUTTON
ICR Type H	FIFO_EMPTY#0, 1, 2, 3, 4, 5, 6, 7
ICR Type I	FIFO_FULL#0, 1, 2, 3, 4, 5, 6, 7

Table 9-6 Input Cell Register Types

Group	*Level*	*Polarity*	*Edge / Level*	*Description*
C1	Fixed Lev#7	Active Low	Level-Sensitive	FAL# ICR has been kept identical to the Rio2 SIC 6351 device for software compatibility.
D	Fixed Lev#7	Active Low	Level-Sensitive	DEG# ICR has been kept identical to the Rio2 SIC 6351 device for software compatibility.
E	Program Lev#7 to 1	Active Low	Edge-Sensitive	Fixed edge-sensitive and active low IT (level is programmable from 7 to 1 (lev#0 = disable)).
EP	Program Lev#7 to 1	Active High	Edge-Sensitive	Fixed edge-sensitive and active high IT (level is programmable from 7 to 1 (lev#0 = disable)).
F	Program Lev#7 to 1	Active Low	Level-Sensitive	Fixed level-sensitive and active low IT (level is programmable from 7 to 1 (lev#0 = disable)).
FE	Program Lev#7 to 1	Active High	Level-Sensitive	Fixed level-sensitive and active high IT (level is programmable from 7 to 1 (lev#0 = disable)).
G	Program Lev#7 to 1	Program	Program	Fully programmable ICR IT.
H	Program Lev#7 to 1	Program	Level-Sensitive	Fixed level-sensitive IT for FIFO empty flags (polarity is programmable for empty/not empty requests).
I	Program Lev#7 to 1	Active High	Program	Fixed active edge high IT (level is programmable from 7 to 1 (lev#0 = disable)).

9.3.4 Interrupt Global Control and Status

Table 9-7 Code Status Register (CSR0)

CPU	*Register Name*
0xF900'0100	CSR0 (read-only)

This register contains the ACT CODE of the last serviced interrupt (0x00 to 0x3F).

Table 9-8 Clear Register (CLR0)

CPU	*Register Name*
0xF900'0104	CLR0 (write-only)

This write-only register is used for debugging purposes only. It can be used to clear the latched edge-sensitive interrupts without performing an IACK cycle to the SIC.

CLR0 can be used to clear the following interrupt sources:

Table 9-9 CLR0 Register Description

	CLR0_Register$_{CPU}$ = 0xF9000'104	
Bit Offset	*IT Source*	*Description*
[07]	FIFO#7_FULL	Clear message passing FIFO#7_FULL flags
[06]	FIFO#6_FULL	Clear message passing FIFO#6_FULL flags
[05]	FIFO#5_FULL	Clear message passing FIFO#5_FULL flags
[04]	FIFO#4_FULL	Clear message passing FIFO#4_FULL flags
[03]	FIFO#3_FULL	Clear message passing FIFO#3_FULL flags
[02]	FIFO#2_FULL	Clear message passing FIFO#2_FULL flags
[01]	FIFO#1_FULL	Clear message passing FIFO#1_FULL flags

Table 9-9 CLR0 Register Description

	CLR0_Register$_{CPU}$ = 0xF9000'104	
Bit Offset	***IT Source***	***Description***
[00]	FIFO#0_FULL	Clear message passing FIFO#0_FULL flags

Table 9-10 Clear Register (CLR1)

CPU	***Register Name***
0xF900'0108	CLR1 (write-only)

This write-only register is used for debugging purposes only. It can be used to clear the latched edge-sensitive interrupts without performing an IACK cycle to the SIC.

CLR1 can be used to clear the following interrupt sources:

Table 9-11 CLR1 Register Description

	CLR1_Register$_{CPU}$ = 0xF9000'108	
Bit Offset	***IT Source***	***Description***
[11]	P0_PCI_PERR	P0 Extended PCI Parity Error signal PERR#
[10]	LOCAL_PCI_PERR	Local PCI Parity Error signal PERR#
[09]	P0_PCI_SERR	P0 Extended PCI system Error signal SERR#
[08]	LOCAL_PCI_SERR	Local PCI System Error signal SERR#
[07]	RS485_IRQ	Front panel RS485 IRQ from µDB9 remote control connector
[06]	TIME_INT#5	Internal TIMER_RAM bit [07]
[05]	TIME_INT#4	Internal TIMER_RAM bit [06]
[04]	TIME_INT#3	Internal TIMER_RAM bit [05]
[03]	TIME_INT#2	Internal TIMER_RAM bit [04]
[02]	TIME_INT#1	Internal TIMER_RAM bit [03]
[01]	TIME_INT#0	Internal TIMER_RAM bit [02]
[00]	TICK_TIMER_IRQ	Internal TIMER_RAM bit [01]

9.3.5 Auto-Mask Register

The auto-mask register allows a global mask of interrupts whose level is below a given value.

When enabled, the level of the interrupt mask is updated, at each IACK cycle, to the value of the last interrupt serviced. Only then, the interrupts with a higher level are allowed to generate interrupts to the CPU.

Table 9-12 Auto-Mask Register (AMSK)

CPU	***Register Name***
0xF900'0200	Auto-Mask (read-only)

AMSK can be used to clear the following interrupt sources:

Table 9-13 AMSK Register Description

	Auto-Mask_Register$_{CPU}$ = 0xF9000'200	
Bit Offset	***Name***	***Description***
[07:04]	ZERO	Always zero (not used by the logic)
[03]	ENA_AUTOMSK	Enables auto-mask level function, when set to 1

Table 9-13 AMSK Register Description

	Auto-Mask_Register$_{CPU}$ = 0xF9000'200	
Bit Offset	***Name***	***Description***
[02:00]	MSK_LEVEL	Contains the value of the updated mask level from the last IACK cycle when the automatic level masking is enabled: 0x000 = all of the interrupts levels are unmasked 0x001 = all of the interrupts on level 7, 6, 5, 4, 3, 2 are unmasked 0x010 = all of the interrupts on level 7, 6, 5, 4, 3 are unmasked 0x011 = all of the interrupts on level 7, 6, 5, 4 are unmasked 0x100 = all of the interrupts on level 7, 6, 5 are unmasked 0x101 = all of the interrupts on level 7, 6 are unmasked 0x110 = all of the interrupts on level 7 are unmasked 0x111 = all of the interrupts levels are masked

9.3.6 Software Interrupt Register (SMIR)

SMI interrupts can be generated to the CPU by writing 1 to bit [00] of the SMIR register.

Table 9-14 Software Interrupt Register (SMIR)

CPU	***Register Name***
0xF900'0300	SMIR

Table 9-15 SMIR Register Description

	Software_Interrupt_Register$_{CPU}$ = 0xF9000'300	
Bit Offset	***Name***	***Description***
[07:01]	Not Used	Always zero
[00]	CPU_SMI	Assert the PowerPC SMI exception, when set to 1

9.3.7 IACK Register (IACK-REG)

Table 9-16 IACK Register (IACK-REG)

CPU	***Register Name***
0xF900'0600	IACK (read-only)

This register provides the interrupt vector of the serviced interrupt sources.

NOTE

The SIC logic provides the vector of the highest interrupt detected.
When no interrupt is pending the vector spurious interrupt is provided.
IACK cycle will automatically clear the edge-sensitive interrupt.

Table 9-17 IACK Interrupt Sources

	IACK_Vector_Register$_{CPU}$ = 0xF9000'600	
Bit Offset	***IT Source***	***Description***
[15:00]	IT_VECTOR	16-bit vector from the VECTOR_SRAM

WARNING

The two upper bits [15:14] of the vector are reserved for the buffer flushing function.

9.3.8 XPC Buffer Flushing

The two upper bits [15:14] of the interrupt vector are used as the flushing pathway:

Table 9-18 XPC Flush Encoding

Vector Bit<15:14>	*Flushing Buffer Path*
[0,0]	No flush is executed on the interrupt
[0:1]	Execute: Local_PCI-to-system memory buffer flushing on the interrupt
[1:0]	Execute: J3_PCI-to-system memory buffer flushing on the interrupt
[1:1]	Execute: CompactPCI-to-system memory buffer flushing on the interrupt

The use of large buffers in the FPGA logics creates a latency time at the end of a DMA transfer between the different XPC_bridges and the system memory. For that, the XPC_CPU FPGA bridge implements a "Buffer-Flush-Done Bit" into the local register LR0, which can be polled to synchronize the processes at the end of a DMA transfer (interrupt) to guarantee the validity of the data transferred into the system memory.

NOTE

The "Buffer-Flush-Done" bit in the XPC_CPU FPGA LR0 register is set to "one" when the transferred data is flushed out of the buffer into the memory.

9.3.9 FPGA Revision Status

The major modifications of the SIC FPGA logics are tagged in this register by incrementing the revision number by one.

Table 9-19 Revision Register Description

	$SIC\_Revision_{CPU}$ = 0xF900'0700		
Bits	***Name***	***Mode***	***Description***
[07:00]	Revision	R-Only	Modified for each major FPGA revision.

9.3.10 Global Interrupt Status

These 32-bits registers show the status of all of the sixty-three interrupt signals at the input of the interrupt controller.

NOTE

Each bit set to "one" in these registers, reflects that the corresponding interrupt request is pending, even if it is masked or disabled in the ICR.

Table 9-20 Global Status Register

CPU	*Register Description*
0xF900'0800	GLB_STAT0: this register monitors the interrupt sources 1 to 32 (1 bit per interrupt pending)
0xF900'0804	GLB_STAT1: this register monitors the interrupt sources 33 to 63 (1 bit per interrupt pending)

Table 9-21 GBL_STAT0 Bit Map

	$GBL\_STAT\_0\ Register_{CPU}$ = 0xF900'0800			
Bit Position	***3***	***2***	***1***	***0***
[31:28]	CPCI_INTA	CPCI_INTB	CPCI_INTC	CPCI_INTD
[27:24]	CPCI_PERR	CPCI_SERR	CPCI_ENUM	LOC_PCI_INTD
[23:20]	LOC_PCI_INTC	LOC_PCI_INTB	PMC2_INTA	PMC1_INTA
[19:16]	SIO_2 (Aux.)	SIO_1 (Cons.)	SPARE (R4K7)	LAN (Ethernet)
[15:12]	FIFO_7_FULL	FIFO_6_FULL	FIFO_5_FULL	FIFO_4_FULL
[11:08]	FIFO_3_FULL	FIFO_2_FULL	FIFO_1_FULL	FIFO_0_FULL
[07:04]	FIFO_7_EMPTY	FIFO_6_EMPTY	FIFO_5_EMPTY	FIFO_4_EMPTY
[03:00]	FIFO_3_EMPTY	FIFO_2_EMPTY	FIFO_1_EMPTY	FIFO_0_EMPTY

NOTE Bit [17] is in spare (ICR18) and it is connected to an external 4K7 pull-up.
Bits [24:22] are connected to both PMCs.

Table 9-22 GBL_STAT1 Bit Map

	GBL_STAT_1 Register$_{CPU}$ = 0xF900'0804			
Bit Position	***3***	***2***	***1***	***0***
[31:28]	Reserved	Reserved	Reserved	Reserved
[27:24]	Reserved	Reserved	J3_PCI_ENUM	J3_PCI_MBX
[23:20]	RS_485	XPC_DMA_IRQ	J3_PCI_SPARE	TIMER_IT5
[19:16]	TIMER_IT4	TIMER_IT3	TIMER_IT2	TIMER_IT1
[15:12]	TIMER_IT0	CPCI_FAL	CPCI_DEG	CPC_SPARE
[11:08]	J3_PCI_SERR	LOC_PCI_SERR	RTC	TIMER
[07:04]	J3_PCI_INTD	J3_PCI_INTC	J3_PCI_INTB	J3_PCI_INTA
[03:00]	THERMOMETERS	J3_PCI_PERR	LOC_PCI_PERR	CPCI_BMA

NOTE Reserved bits are always read as "zero".

EXAMPLE Bit [18] set to 1 of GLB_STAT0, means that an interrupt corresponding to interrupt source SIO#1 console, is pending (ICR19).

9.3.11 Global Interrupt Mask

These 32-bits registers authorize to enable or disable all of the sixty-three interrupts into one or two access, instead of accessing all the sixty-three ICRs.

NOTE Each bit set to 1 enable the corresponding interrupt, each bit set to 0 disable the corresponding interrupt.

Table 9-23 Global Mask Register

CPU	***Register Description***
0xF900'0808	GLB_MSK0: this register enables / disables the interrupt sources 1 to 32 (1 bit per interrupt pending)
0xF900'080C	GLB_MSK1: this register enables / disables the interrupt sources 33 to 63 (1 bit per interrupt pending)

Table 9-24 GBL_MSK0 Bit Map

	GBL_MSK_0 Register$_{CPU}$ = 0xF900'0808			
Bit Position	***3***	***2***	***1***	***0***
[31:28]	CPCI_INTA (ICR_32)	CPCI_INTB (ICR_31)	CPCI_INTC (ICR_30)	CPCI_INTD (ICR_29)
[27:24]	CPCI_PERR (ICR_28)	CPCI_SERR (ICR_27)	CPCI_ENUM (ICR_26)	LOC_PCI_INTD (ICR_25)
[23:20]	LOC_PCI_INTC (ICR_24)	LOC_PCI_INTB (ICR_23)	PMC2_INTA (ICR_22)	PMC1_INTA (ICR_21)
[19:16]	SIO_2 (Ax.) (ICR_20)	SIO_1 (Cons.) (ICR_19)	SPARE (R4K7) (ICR_18)	LAN (Ethernet) (ICR_17)
[15:12]	FIFO_7_FULL (ICR_16)	FIFO_6_FULL (ICR_15)	FIFO_5_FULL (ICR_14)	FIFO_4_FULL (ICR_13)
[11:08]	FIFO_3_FULL (ICR_12)	FIFO_2_FULL (ICR_11)	FIFO_1_FULL (ICR_10)	FIFO_0_FULL (ICR_9)
[07:04]	FIFO_7_EMPTY (ICR_8)	FIFO_6_EMPTY (ICR_7)	FIFO_5_EMPTY (ICR_6)	FIFO_4_EMPTY (ICR_5)

Table 9-24 GBL_MSK0 Bit Map

	GBL_MSK_0 Register$_{CPU}$ = 0xF900'0808			
Bit Position	***3***	***2***	***1***	***0***
[03:00]	FIFO_3_EMPTY (ICR_4)	FIFO_2_EMPTY (ICR_3)	FIFO_1_EMPTY (ICR_2)	FIFO_0_EMPTY (ICR_1)

EXAMPLE

Setting bit [18] of GLB_MSK0 enables the interrupt source SIO#1 console.
Setting or clearing a bit overwrites bit [07] (enable / disable) of the corresponding ICR_19.

Table 9-25 GBL_MSK1 Bit Map

	GBL_MSK_1 Register$_{CPU}$ = 0xF900'080C			
Bit Position	***3***	***2***	***1***	***0***
[31:28]	Reserved	Reserved	Reserved	Reserved
[27:24[	Reserved	Reserved	J3_PCI_ENUM (ICR_58)	J3_PCI_MBX (ICR_57)
[23:20]	RS_485 (ICR_56)	XPC_DMA_IRQ (ICR_55)	J3_PCI_SPARE (ICR_54)	TIMER_IT5 (ICR_53)
[19:16]	TIMER_IT4 (ICR_52)	TIMER_IT3 (ICR_51)	TIMER_IT2 (ICR_50)	TIMER_IT1 (ICR_49)
[15:12]	TIMER_IT0 (ICR_48)	CPCI_FAL (ICR_47)	CPCI_DEG (ICR_46)	CPCI_SPARE (ICR_45)
[11:8]	J3_PCI_SERR (ICR_44)	LOC_PCI_SERR (ICR_43)	RTC (ICR_42)	TIMER (ICR_41)
[7:4]	J3_PCI_INTD (ICR_40)	J3_PCI_INTC (ICR_39)	J3_PCI_INTB (ICR_38)	J3_PCI_INTA (ICR_37)
[3:0]	THERMOMETERS (ICR_36)	J3_PCI_PERR (ICR_35)	LOC_PCI_PERR (ICR_34)	CPCI_BMA (ICR_33)

EXAMPLE

Setting bit [22] of GLB_MSK1 enables the interrupt source XPC_DMA_IRQ. Setting or clearing a bit overwrites bit [07] (enable / disable) of the corresponding ICR_55.

Chapter 10
DMA Controller

10.1 DMA Block Diagram Description

A multichannel, multipriority, chained DMA engine is embedded in the PowerPC-to-XPC bridge. This DMA engine is able to move data blocks autonomously from the system memory to any XPC agent and also from CompactPCI to any XPC agent.

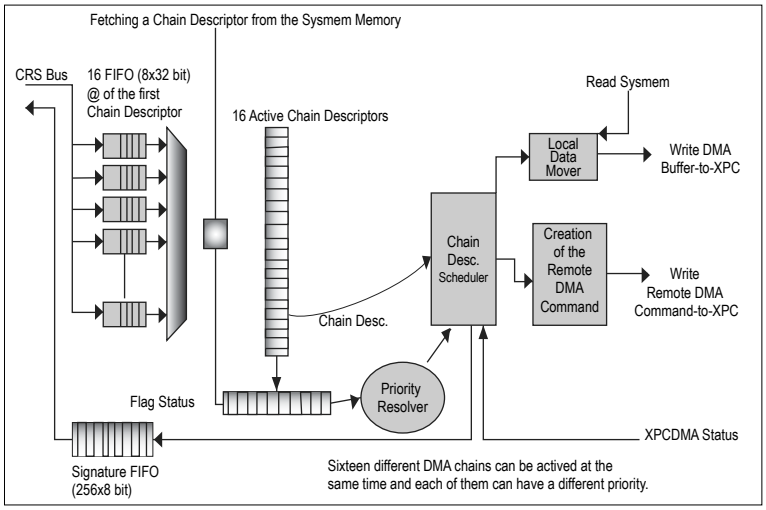
- XPC-PowerPC bridge embedded DMA controller
- 16 virtual channels
- Chained DMA
- Local system memory to any XPC
- Remote DMA read engine (only available in the XPC_CompactPCI bridge)

The data block transfers are controlled by a "chain" of individual descriptors. The CPU builds these chains in the system memory and triggers the DMA controller by writing the address of the first descriptor in one of the start address FIFOs.

By using this address, the DMA reads the full descriptor from the system memory. The necessary number of elementary data move operations will be performed until the entire data transfer has been completed. Then, if the end of the chain has not been reached, the DMA engine will read the next chain descriptor from the system memory. If the end of chain has been reached, the DMA engine will write a "DMA_signature" in a FIFO to signal the end of the chain to the PowerPC.

Sixteen channels are available and each of them are given a different priority (please refer to section 10.2.3). All of these 16 channels can run at the same time.

Figure 10.1 DMA Block Diagram



NOTE Empty or overflow signals are only available at the hardware level for the Signature_FIFO or for the start of chain FIFOs.

10.2 DMA Chaining Element

The DMA controller is able to move large data blocks without consuming any CPU time. Large data blocks must be cut in smaller ones. This is done by building a chain of smaller data blocks to be moved. This chaining also provides the capability to build DMA with scatter / gather. (UNIX 4 KBytes pages).

Each of these smaller data blocks are described by a "chaining element" in the DMA chain. A chaining element is a set of four 64-bit wide words. Two different formats are used (“CompactPCI_Write” and “CompactPCI_Read”).

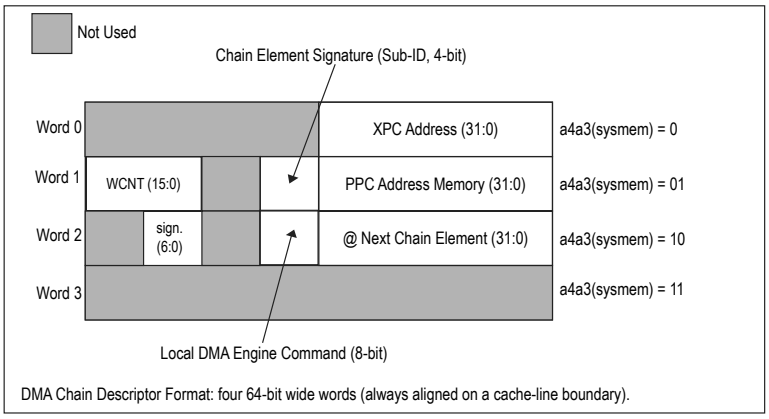
10.2.1 DMA Chaining Element for “CompactPCI_Write”

“CompactPCI_Write” is used for data transfer from the system memory to one of the three XPC bridges. Because the DMA engine is mainly used to transfer data from system memory to the CompactPCI backplane, it is called “CompactPCI_Write”. Although, the destination may also be the PCI local bus or the PCI_J3 bus.

The source address must be in the system memory.

The following chaining element format must be used for data move transfers from system memory to the XPC bus:

Figure 10.2 CompactPCI_Write Chaining Element



DMA Chain Descriptor Format: four 64-bit wide words (always aligned on a cache-line boundary).

Table 10-1 Chaining Element WORD_0

Bits	*Name*	*Mode*	*Description*
[63:32]	Not Used	-	Reserved
[31:00]	XPC_W_Address	-	XPC "WRITE to" address (can not be system memory address)

Table 10-2 Chaining Element WORD_1

Bits	*Name*	*Mode*	*Description*
[63:48]	CE_WCNT	-	16-bit word counter: 1 to (64k-1) 32-bit words to be transferred
[47:40]	Not Used	-	Reserved
[39:36]	dmaSUBID	-	DMA second signature
[35:32]	Not Used	-	Reserved
[31:00]	System Memory READ Address	-	System memory "READ from" address

Table 10-3 Chaining Element WORD_2

Bits	*Name*	*Mode*	*Description*
[63:55]	Not Used	-	Reserved
[54:48]	Signature	-	DMA chain element signature
[47:40]	Not Used	-	Reserved
[39]	CPCI_READ_Remote	-	Must be cleared to 0

Table 10-3 Chaining Element WORD_2

Bits	Name	Mode	Description
[38:37]	CPCI_Boundary	-	This field allows the user to limit the address boundary crossing: 00 : no boundary crossing constraint 01 : 256 Bytes boundary 10 : 2k Bytes boundary 11 : 4k Bytes boundary
[36]	DoubleBUF	-	Use "Double Size Buffer" for the XPC bus transport. Normally the XPC bus uses write-buffer of 256 Bytes. Two of them can be merged to form 512 Bytes buffer ("Double Size Buffer")
[35:34]	Not Used	-	Reserved
[33]	ChainSNOOP	-	When set, the PowerPC bus snooping is enabled to get data from the system memory (DATA cache coherency control)
[32]	FetchNEXT	-	When 1, at end of current chaining element, fetch the next chaining element pointed by the field [29:4] When 0, last chaining element, end of chain
[31:30]	NextCHAIN_ADD[31:30]	-	Must be set to 00
[29:04]	NextCHAIN_ADD[29:04]	-	Address in system memory for next chaining element to be fetched by the scheduler
[03:00]	NextCHAIN_ADD[03:00]	-	Must be set to 0000

Table 10-4 Chaining Element WORD_3

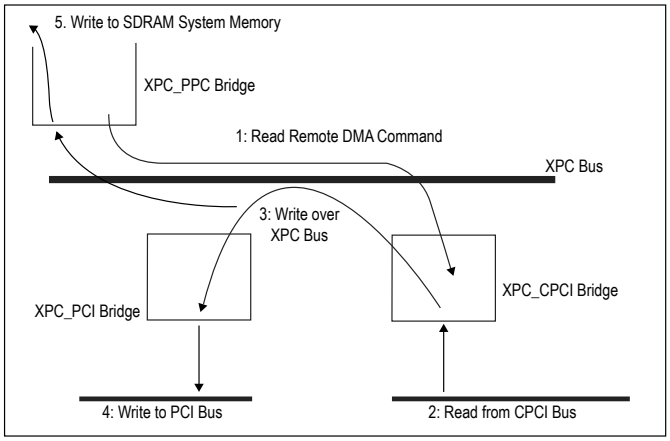
Bits	Name	Mode	Description
[63:00]	Not Used	-	Reserved

10.2.2 DMA Chaining Element for XPC "CompactPCI_Read"

CompactPCI_Read is used for the data transfer from the CompactPCI bus to one of the three bridges on the XPC bus. Only the CompactPCI_Read operation is supported due to the fact that the XPC_LocPCI and XPC_PCIJ3 bridges do not incorporate the remote read logic.

To provide better performance, the CompactPCI_Read machine is embedded in the XPC_CompactPCI bridge FPGA. This allows to use XPC_Write cycle, which provides maximum XPC bandwidth capability. This remote state-machine is triggered by the DMA controller with a "read remote DMA command". While triggered, the remote state-machine executes the command (read n word…) and sets-up a DMA status handshake with the DMA controller.

Figure 10.3 Remote CompactPCI_READ Sequencing



The following chaining element format must be used for data move transfers from CompactPCI.

Figure 10.4 Compact_READ Chaining Element

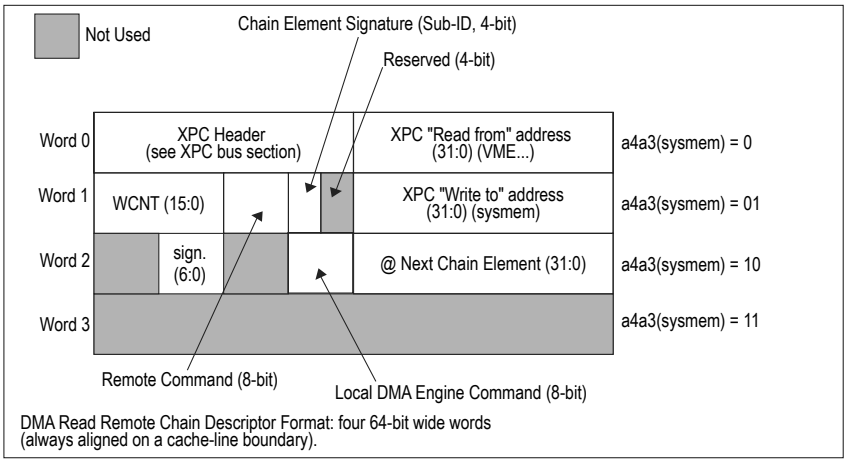


Table 10-5 Chaining Element WORD_0

Bits	*Name*	*Mode*	*Description*
[63:62]>	XPC_Header_TDST	-	Should be = 11 targeting CompactPCI bridge
[61:60]	XPC_Header_TSRC	-	Must be = 00
[59:58]	XPC_Header_TTYPE	-	Must be = 00 for memory space
[57]	XPC_Header_SNOOP	-	Must be = 0, snoop not used
[56]	XPC_Header_TWRITE	-	Must be = 1
[55]	XPC_Header_TBURST	-	Must be = 1, for single cycle
[54:52]	XPC_Header_TSIZ	-	Must be = 000, for 8 Bytes write
[51:50]	XPC_Header_APM	-	Must be = 10, for Type_1
[49]	XPC_Header_RFLUSH	-	Must be = 0, not used
[48]	XPC_Header_DSIZ	-	Must be = 0, not used
[47:40]	XPC_Header_TCNT	-	Must be = 00000001, for single beat
[39:32]	XPC_Header_EAD	-	Not used
[31:00]	XPC_R_Address	-	XPC "READ from" address (CompactPCI space)

NOTE Bits [63:32] (XPC_Header) should hold an emulation of the XPC_Header section.

Table 10-6 Chaining Element WORD_1

Bits	*Name*	*Mode*	*Description*
[63:48]	CE_WCNT	-	16-bit word counter: 1 to (64k-1) 32-bit words to be transferred
[47:46]	CPCI_MaxSIZ	-	Maximum CompactPCI read burst size 11 : 64 words 10 : 32 words 01 : 15 words 00 : 8 words
[45]	CPCI_NoINC	-	While set, the CompactPCIaddress is not incremented during the DMA transfer. Used to read-out FIFO-based CompactPCI slave interface.

Table 10-6 Chaining Element WORD_1

Bits	Name	Mode	Description
[44:43]	dmaXPC_DST	-	XPC bridge destination, can be different from 00 (system memory) 00 : System memory 01 : Local PCI 10 : PCI J3 (External PCI)
[42]	dmaXPC_TYPE	-	XPC address type 0 : Memory space 1 : I/O space
[41]	dmaXPC_SNOOP	-	XPC write with cache coherency control 0 : Snoop disabled 1 : Snoop enabled
[40]	dmaDSIZ_ENA	-	XPC double size buffer use enabled 0 : Can not use double size buffer 1 : Can use double size buffer
[39:36]	dmaSUBID	-	DMA second signature
[35:32]	Not Used	-	Reserved
[31:00]	XPC_W_Address	-	XPC "WRITE to" address

Table 10-7 Chaining Element WORD_2

Bits	Name	Mode	Description
[63:55]	Not Used	-	Reserved
[54:48]	Signature	-	DMA chain element signature
[47:40]	Not Used	-	Reserved
[39]	CPCI_READ_Remote	-	Must be set to 1
[38:37]	Not Used	-	Reserved
[36]	Not Used	-	Reserved
[35:34]	Not Used	-	Reserved
[33]	Not Used	-	Reserved
[32]	FetchNEXT	-	When 1, at end of current chaining element, fetch the next chaining element pointed by the field [29:4] When 0, last chaining element, end of chain
[31:30]	NextCHAIN_ADD[31:30]	-	Must be set to 00
[29:04]	NextCHAIN_ADD[29:04]	-	Address in system memory for next chaining element to be fetched by the scheduler
[03:00]	NextCHAIN_ADD[03:00]	-	Must be set to 0000

Table 10-8 Chaining Element WORD_3

Bits	Name	Mode	Description
[63:00]	Not Used	-	Reserved

Two lines are used to indicate the state of the read remote DMA to the central DMA scheduler. These two lines are routed from the XPC_CompactPCI bridge to the DMA controller, located in the XPC_PPC bridge.

NOTE These two sideband signals can not be read by software.

10.2.3 DMA Priority Mechanism

The DMA controller can manage up to 16 channels simultaneously. Each virtual channel owns a private 8 FIFO deep FIFO entry for its "start of chain" address. A priority mechanism is implemented to control the QoS (Quality of Service). The priority

rescheduling is handled at the chaining element level, therefore the chaining element burst size (1 to 64K-1 words) will determine the DMA priority rescheduling latency.

- Eight channels (channel 15 to 8) are handled in a fixed priority mechanism. The channel with the highest priority is executed. If a channel with a higher priority is started, the DMA scheduler will switch to this channel at the next chaining element fetch. Channel 15 has the highest priority and channel 8 the lowest.

- Eight channels (channel 7 to 0) are handled in a round-robin type mechanism. They are scruted in a circular loop one after the other. Since they have the same relative priority, the channel rescheduling is done at the chaining element granularity. These eight channels have a lower priority than channel 8.

The DMA channel allocation and the chaining element word size will have a strong impact on the expected QoS.

10.2.4 DMA Normal End of Transfer

Once a DMA chain has been completely processed by the DMA controller, the DMA signature, held by the last executed chaining element, is stored in the Signature_FIFO. The Not_EMPTY state of this FIFO can issue an interrupt to the PowerPC CPU. This function will be enabled with DMA_IRQENA_EF located in the XPCPPC_CTL register.
To complete the handling, the related interrupt handler will read the XPCPPC_DMASIGN register, which stores the DMA signature.

10.2.5 DMA End of Transfer Error

The DMA process is aborted in case of error detection.

The DMA process “Compact_Read” is directly linked with CompactPCI error detection logic. In the case of a CompactPCI error detection, the DMA chain is aborted and the DMA_Signature is loaded in the Signature_FIFO with bit “DMA_Sign_ERR“ set. It is also possible to enable a special DMA_Error interrupt, enabled with ‘’DMA_IRQENA_RERR” located in the ”XPCPPC_CTL” register.

Since all DMA transfer “CompactPCI_Write” use write posting capability, the DMA controller has no feedback information from the CompactPCI master and therefore no capability to abort the chain. In this case, the error recovery must be handled with other error tracking mechanisms provided with:
- XPCPPC_XERR register in case of XPC error.

10.3 DMA Control Registers

The DMA controller is interfaced through two register sets.

Table 10-9 DMA Controller Registers Mapping

Access from CPU	*Registers*	*Description*
0xF900’9340	XPCPPC_DMASIGN	End of DMA chain signature. Read port of FIFO.
0xF900’93C0	XPCPPC_DMASTART#0	DMA channel #0 address of the first chain descriptor. FIFO 8 words deep.
0xF900’93C4	XPCPPC_DMASTART#1	Start of DMA’s chain channel #1.
...	...	....
0xF900’93FC	XPCPPC_DMASTART#15	Start of DMA’s chain channel #15.

NOTE

Since these registers are mapped over the XCSR bus, they can also be accessed from agents other than the local PowerPC CPU.

XPCPPC_DMA_START

The sixteen XPCPPC_DMASTARTs registers allow to trigger the associated DMA’s chain. Behind each of these registers, an 8-word deep FIFO is implemented, allowing to pipeline up to eight start-of-chain per virtual channel. These registers accept a PowerPC physical address, which target the onboard system memory. Any other address range can cause unpredictable results.

Table 10-10 XPCPPC_DMA_Start

Bits	*Name*	*Mode*	*Description*
[31:30]	DMA_StartADD[31:30]	W	Must be “00”
[29:04]	DMA_StartADD[29:04]	W	System memory physical address
[03:00]	DMA_StartADD[03:00]	W	Must be “0000”. The start address is aligned on a cache-line boundary

XPCPPC_DMASIGN

This register allows to access the 128 word deep FIFO holding the end of transfer signalling.
At the end of a DMA transfer, the chain signature is written in this FIFO to signal the end of the DMA transfer for this particular chain. The not empty flag of this FIFO is used to issue a local interrupt.

Table 10-11 XPCPPC_DMA_Sign

	PPC_CTLCPU = 0xF900’9340		
Bits	*Name*	*Mode*	*Description*
[31:16]	Not Used	R	Reserved
[15]	DMA_Sign_EF	R	DMA signature FIFO not empty flag. This bit is duplicated here for debugging purposes. An incorrect signature will be read out from an empty FIFO, so an incorrect signature will have its bit 15 set to one.
[14:12]	Not Used	R	Read as “000”
[11:08]	DMA_Sign_SubID	R	Sub-ID identifier in the last chain element in the DM, located on bit [39:36] chain. Debug tracking of the last chain element processed by the DMA engine. In case of error, the DMA engine will abort the processing of the chain, and copy the Sub-ID field of the chain element as default.
[07]	DMA_Sign_ERR	R	When set to "1", this bit indicates that the DMA engine met an error while processing this chain (only in the case of “CompactPCI_Read”).
[06:00]	DMA_Sign_ID	R	This 7-bit field is written in the signature FIFO at the end of the processing of a DMA chain, so the software knows which DMA chain has been completed. These 7 bits are in word two chain element, on bit [54:48].

NOTE

No hardware overflow management is performed. Care must be taken at the software level to not let an overflow occur. No more than a maximum of sixteen start-of-chain in the system are allowed at a given time.

Chapter 11
Annexes

11.1 CompactPCI Connectors

11.1.1 CompactPCI J1 Connector

Table 11-1 CompactPCI J1 Connector

Pin	*A*	*B*	*C*	*D*	*E*
25	5 V	REQ64#	ENUM#	3.3 V	5 V
24	AD1	5 V	VI/O_PC	AD0	ACK64#
23	3.3 V	AD4	AD3	5 V_PC	AD2
22	AD7	GND	3.3 V_PC	AD6	AD5
21	3.3 V	AD9	AD8	M66EN# (10K)	CBE0
20	AD12	GND	VI/O	AD11	AD10
19	3.3 V	AD15	AD14	GND_PC	AD13
18	SERR#	GND	3.3 V	PAR	CBE1
17	3.3 V	SDONE#	SBO#	GND_PC	PERR#
16	DEVSEL#	GND	VI/O	STOP#	LOCK#
15	3.3 V	FRAME#	IRDY#	BDSEL#	TRDY#
12-14	KEY	KEY	KEY	KEY	KEY
11	AD18	AD17	AD16	GND_PC	CBE2
10	AD21	GND	3.3 V	AD20	AD19
9	CBE3	IDSEL	AD23	GND_PC	AD22
8	AD26	GND	VI/O	AD25	AD24
7	AD30	AD29	AD28	GND_PC	AD27
6	REQ#	GND	3.3 V_PC	CLK	AD31
5	NC	NC	RST#	GND_PC	GNT#
4	NC	HEALTHY#	VI/O_PC	NC	NC
3	INTA#	INTB#	INTC#	5 V_PC	INTD#
2	NC	5 V	NC	JTAG_TDO	JTAG_TDI
1	5 V	- 12 V	NC	+ 12 V	5 V

NOTE

JTAG_TDI is wired to JTAG_TDO through a 10 ohm resistor onto the PCB. Other JTAG signals are not connected (NC).

11.1.2 CompactPCI J2 Connector

Table 11-2 CompactPCI J2 Connector

Pin	*A*	*B*	*C*	*D*	*E*
22	GA4	GA3	GA2	GA1	GA0
21	CLK6	NC	NC	NC	NC
20	CLK5	NC	NC	NC	NC
19	NC	NC	NC	NC	NC
18	NC	NC	NC	NC	NC
17	NC	GND	PRST#	REQ6#	CBE4
16	NC	NC	DEG#	GND	NC
15	NC	GND	FAL#	REQ5#	GNT5#
14	AD35	AD34	AD33	GND	AD32
13	AD38	GND	VI/O	AD37	AD36
12	AD42	AD41	AD40	GND	AD39
11	AD45	GND	VI/0	AD44	AD43
10	AD49	AD48	AD47	GND	AD46
9	AD52	GND	VI/O	AD51	AD50
8	AD56	AD55	AD54	GND	AD53
7	AD59	GND	VI/O	AD58	AD57
6	AD63	AD62	AD61	GND	AD60
5	CBE5#	GND	VI/O	CBE4#	PAR64
4	VI/O	NC	CBE7#	GND	CBE6#
3	CLK4	GND	GNT3#	REQ4#	GNT4#
2	CLK2	CLK3	SYSEN#	GNT2#	REQ3#
1	CLK1	GND	REQ1#	GNT1#	REQ2#

11.1.3 J3: PCI-EXT

Table 11-3 J3: PCI-EXT

Pin	*A*	*B*	*C*	*D*	*E*
19	REQ#	IDSEL	REQ64#	STOP#	FRAME#
18	GNT#	LOCK#	RST#	ACK64#	DEVSEL#
17	PAR64	CBE7	CBE6	CBE5	CBE4
16	AD36	AD35	AD34	AD33	AD32
15	AD41	AD40	AD39	AD38	AD37

Table 11-3 J3: PCI-EXT

Pin	*A*	*B*	*C*	*D*	*E*
14	AD46	AD45	AD44	AD43	AD42
13	AD51	AD50	AD49	AD48	AD47
12	AD56	AD55	AD54	AD53	AD52
11	AD61	AD60	AD59	AD58	AD57
10	AD0	TRDY#	IRDY#	AD63	AD62
9	AD5	AD4	AD3	AD2	AD1
8	AD10	AD9	AD8	AD7	AD6
7	AD15	AD14	AD13	AD12	AD11
6	AD20	AD19	AD18	AD17	AD16
5	AD25	AD24	AD23	AD22	AD21
4	AD30	AD29	AD28	AD27	AD26
3	CBE3	CBE2	CBE1	CBE0	AD31
2	SPARE	RESET#	SYNC	CLK_IN	PAR
1	PRESENT#	SEROUT	SERIN	CLK_SRC#2	CLK_SRC#1

11.1.4 CompactPCI J4 Connector

Table 11-4 CompactPCI J4 Connector

Pin	*A*	*B*	*C*	*D*	*E*
25	RS485_TrigNeg	Rem_RST_Anod	NC	RS485_Dat1_Neg	RS485_Dat2_Neg
24	RS485_TrigPos	Rem_RST_Catod	NC	RS485_Dat1_Pos	RS485_Dat2_Pos
23	NC	NC	NC	NC	NC
22	NC	NC	NC	NC	NC
21	NC	NC	NC	NC	NC
20	NC	NC	NC	NC	NC
19	NC	NC	NC	NC	NC
18	NC	NC	NC	NC	NC
17	+ 3.3 V	+ 3.3 V	+ 3.3 V	+ 5 V	+5 V
16	JN24-33	JN24-34	JN24-35	JN24-36	JN24-37
15	JN24-38	JN24-39	JN24-40	JN24-41	JN24-42
12-14	KEY	KEY	KEY	KEY	KEY
11	JN24-43	JN24-44	JN24-45	JN24-46	JN24-47
10	JN24-48	JN24-49	JN24-50	JN24-51	JN24-52
9	JN24-53	JN24-54	JN24-55	JN24-56	JN24-57
8	JN24-58	JN24-59	JN24-60	JN24-61	JN24-62
7	JN24-63	JN24-64	JN14-33	JN14-34	JN14-35
6	JN14-36	JN14-37	JN14-38	JN14-39	JN14-40
5	JN14-41	JN14-42	JN14-43	JN14-44	JN14-45

Table 11-4 CompactPCI J4 Connector

Pin	*A*	*B*	*C*	*D*	*E*
4	JN14-46	JN14-47	JN14-48	JN14-49	JN14-50
3	JN14-51	JN14-52	JN14-53	JN14-54	JN14-55
2	JN14-56	JN14-57	JN14-58	JN14-59	JN14-60
1	JN14-61	JN14-62	JN14-63	JN14-64	NC

NOTE

J4 BACK_I/O +5V and 3.3V power are connected through an electronic fuse (poly-switch) for a maximum of 750 mA for each 3.3 V and + 5 V power supply.
Rem_RST_Anode pin (B25) is connected to the Opto-copler (HCPL 070A) trough 10K ohm resistor to limit the current RS485 input/output (Trig, Dat1, Dat2) are directly connected to the driver (TI 75LB176).

WARNING

The 32 bits of JN14-33 to JN14-64 are connected with JN24-1 to JN24-32 on the J4 BACK_I/O connector to comply with the previous CES products

11.1.5 CompactPCI J5 Connector

Table 11-5 CompactPCI J5 Connector

Pin	*A*	*B*	*C*	*D*	*E*
22	NC	NC	NC	NC	NC
21	NC	NC	NC	NC	NC
20	NC	NC	NC	NC	NC
19	NC	NC	NC	NC	NC
18	NC	NC	NC	NC	NC
17	NC	NC	NC	NC	NC
16	NC	NC	NC	NC	NC
15	NC	NC	NC	NC	NC
14	NC	NC	NC	NC	NC
13	NC	NC	NC	NC	NC
12	NC	NC	NC	NC	NC
11	NC	NC	NC	NC	NC
10	NC	NC	NC	NC	NC
9	NC	NC	NC	NC	NC
8	NC	NC	NC	NC	NC
7	NC	NC	NC	NC	NC
6	NC	NC	NC	NC	NC
5	NC	NC	NC	NC	NC
4	NC	NC	NC	NC	NC
3	NC	NC	NC	NC	NC
2	NC	NC	NC	NC	NC
1	NC	NC	NC	NC	NC

11.2 PMC Connectors

11.2.1 Jn11 / Jn21 PMC PCI Connectors

Table 11-6 Jn11 / Jn21 PMC PCI Connectors

Signal Name	*Jn11 / Jn21 PIN*	*Signal Name*	*Jn11 / Jn21 PIN*
JTAG_TCK	1	- 12 V	2
GND	3	INTA#	4
INTB#	5	INTC#	6
BUSMODE1	7	+ 5 V	8
INTD#	9	NC	10
GND	11	NC	12
CLK	13	GND	14
GND	15	GNT#	16
REQ#	17	+ 5 V	18
3 V 3 (VIO)	19	AD31	20
AD28	21	AD27	22
AD25	23	GND	24
GND	25	CBE3	26
AD22	27	AD21	28
AD19	29	+ 5 V	30
3 V 3 (VIO)	31	AD17	32
FRAME#	33	GND	34
GND	35	IRDY#	36
DEVSEL#	37	+ 5 V	38
GND	39	LOCK#	40
SDONE# (4K7)	41	SBO# (4K7)	42
PAR	43	GND	44
3 V 3 (VIO)	45	AD15	46
AD12	47	AD11	48
AD9	49	+ 5 V	50
GND	51	CBE0	52
AD6	53	AD5	54
AD4	55	GND	56
3 V 3 (VIO)	57	AD3	58
AD2	59	AD1	60
AD0	61	+ 5 V	62
GND	63	REQ64#	64

11.2.2 Jn12 / Jn22 PMC PCI Connectors

Table 11-7 Jn12 / Jn22 PMC PCI Connectors

Signal Name	*Jn12 / Jn22 PIN*	*Signal Name*	*Jn12 / Jn22 PIN*
+ 12 V	1	JTAG_TRST#	2
JTAG_TMS	3	JTAG_TDO	4
JTAG_TDI	5	GND	6
GND	7	NC	8
NC	9	NC	10
BUSMODE2	11	3.3 V	12
RST#	13	BUSMODE3	14
3.3 V	15	BUSMODE4	16
NC	17	GND	18
AD30	19	AD29	20
GND	21	AD26	22
AD24	23	3.3 V	24
IDSEL	25	AD23	26
3.3 V	27	AD20	28
AD18	29	GND	30
AD16	31	CBE2	32
GND	33	IDSEL_2	34
TRDY#	35	3.3 V	36
GND	37	STOP#	38
PERR#	39	GND	40
3.3 V	41	SERR#	42
CBE1	43	GND	44
AD14	45	AD13	46
GND	47	AD10	48
AD8	49	3.3 V	50
AD7	51	REQ_2#	52
3.3 V	53	BG_2#	54
NC	55	GND	56
NC	57	EREADY	58
GND	59	RSTOUT#	60
ACK 64#	61	3.3 V	62
GND	63	MONARCH#	64

11.2.3 Jn13 / Jn23 PMC (PCI 64 Bits) Connectors

Table 11-8 Jn13 / Jn23 PMC (PCI 64 Bits) Connectors

Signal Name	*Jn13 / Jn23 PIN*	*Signal Name*	*Jn13 / Jn23 PIN*
NC	1	GND	2
GND	3	CBE7#	4
CBE6#	5	CBE5#	6
CBE4#	7	GND	8
3 V 3 (VIO)	9	PAR64	10
AD55	11	AD62	12
AD61	13	GND	14
GND	15	AD60	16
AD59	17	AD58	18
AD57	19	GND	20
3 V 3 (VIO)	21	AD56	22
AD55	23	AD54	24
AD49	25	GND	26
GND	27	AD52	28
AD39	29	AD50	30
AD49	31	GND	32
GND	33	AD48	34
AD47	35	AD46	36
AD45	37	GND	38
3.3 V (VIO)	39	AD44	40
AD43	41	AD42	42
AD41	43	GND	44
GND	45	AD40	46
AD39	47	AD38	48
AD37	49	GND	50
GND	51	AD36	52
AD35	53	AD34	54
AD33	55	GND	56
3.3 V (VIO)	57	AD32	58
NC	59	NC	60
NC	61	GND	62

11.2.4 Jn24 PMC 2 User I/O-to-CompactPCI J4 Connectors

Table 11-9 Jn24 PMC 2 User I/O-to-CompactPCI J4 Connectors

Signal Name	*Jn24 PIN*	*CPCI J4*	*Signal Name*	*Jn24 PIN*	*CPCI J4*
User Defined	1	C7	User Defined	2	D7
User Defined	3	E7	User Defined (*)	4	A6
User Defined	5	B6	User Defined	6	C6
User Defined	7	D6	User Defined	8	E6
User Defined	9	A5	User Defined	10	B5
User Defined	11	C5	User Defined	12	D5
User Defined	13	E5	User Defined	14	A4
User Defined	15	B4	User Defined	16	C4
User Defined	17	D4	User Defined	18	E4
User Defined	19	A3	User Defined	20	B3
User Defined	21	C3	User Defined	22	D3
User Defined	23	E3	User Defined	24	A2
User Defined	25	B2	User Defined	26	C2
User Defined	27	D2	User Defined	28	E2
User Defined	29	A1	User Defined	30	B1
User Defined	31	C1	User Defined	32	D1
User Defined	33	A16	User Defined	34	B16
User Defined	35	C16	User Defined	36	D16
User Defined	37	E16	User Defined	38	A15
User Defined	39	B15	User Defined	40	C15
User Defined	41	D15	User Defined	42	E15
User Defined	43	A11	User Defined	44	B11
User Defined	45	C11	User Defined	46	D11
User Defined	47	E10	User Defined	48	A10
User Defined	49	B10	User Defined	50	C10
User Defined	51	D10	User Defined	52	E10
User Defined	53	A9	User Defined	54	B9
User Defined	55	C9	User Defined	56	D9
User Defined	57	E9	User Defined	58	A8
User Defined	59	B8	User Defined	60	C8
User Defined	61	D8	User Defined	62	E8
User Defined	63	A7	User Defined	64	B7

NOTE

Only the Lower 32 bits are connected to the J4 User I/O.
(*) The ATM SYNCHRO_PULSE is connected to JN24-pin4 and JN14-pin36.

11.2.5 Jn14 PMC 1 User I/O-to-CompactPCI J4 Connectors

Table 11-10 Jn14 PMC 1 User I/O-to-CompactPCI J4 Connectors

Signal Name	*Jn14 PIN*	*CPCI J4*	*Signal Name*	*Jn14 PIN*	*CPCI J4*
User Defined	1	NC	User Defined	2	NC
User Defined	3	NC	User Defined	4	NC
User Defined	5	NC	User Defined	6	NC
User Defined	7	NC	User Defined	8	NC
User Defined	9	NC	User Defined	10	NC
User Defined	11	NC	User Defined	12	NC
User Defined	13	NC	User Defined	14	NC
User Defined	15	NC	User Defined	16	NC
User Defined	17	NC	User Defined	18	NC
User Defined	19	NC	User Defined	20	NC
User Defined	21	NC	User Defined	22	NC
User Defined	23	NC	User Defined	24	NC
User Defined	25	NC	User Defined	26	NC
User Defined	27	NC	User Defined	28	NC
User Defined	29	NC	User Defined	30	NC
User Defined	31	NC	User Defined	32	NC
User Defined	33	C7	User Defined	34	D7
User Defined	35	E7	User Defined (*)	36	A6
User Defined	37	B6	User Defined	38	C6
User Defined	39	D6	User Defined	40	E6
User Defined	41	A5	User Defined	42	B5
User Defined	43	C5	User Defined	44	D5
User Defined	45	E5	User Defined	46	A4
User Defined	47	B4	User Defined	48	C4
User Defined	49	D4	User Defined	50	E4
User Defined	51	A3	User Defined	52	B3
User Defined	53	C3	User Defined	54	D3
User Defined	55	E3	User Defined	56	A2
User Defined	57	B2	User Defined	58	C2
User Defined	59	D2	User Defined	60	E2
User Defined	61	A1	User Defined	62	B1
User Defined	63	C1	User Defined	64	D1

11.3 RS232 Connectors

Table 11-11 RS232 Connectors

Signal Name	*Pin*	*Signal Name*	*Pin*
-		CD	1
DSR	6	RX	2
RTS	7	TX	3
CTS	8	DTR	4
R1	9	GND	5

NOTE

The pin-out is the same for console and auxiliary connectors.
ITT CANON Connector Type MDSM-18PE-Z21VR25SPL.
Metallic part of the connector is connected to the shield.

11.4 RS485 Remote Connectors

These signals are connected to the CPCI J4 back_io connector. For more information, refer to table 11-4.

11.5 Debugging and Setup Connectors

Table 11-12 CISP

Signal Name	*Pin*	*Signal Name*	*Pin*
VCC	A1	PRESENT#	B1
GND	A2	FLASH#	B2
ISPLSI#	A3	cesTRST#	B3
cesMODE	A4	cesSCLK	B4
cesSDO	A5	cesSDI	B5
SPARE_1	A6	SPARE_2	B6

NOTE

SPARE_1 and SPARE_2 are not connected.
Connector male 12p.90 Deg Type: ERNI 054594 - SMD.

Table 11-13 PMC-JTAG

Signal Name	*Pin*	*Signal Name*	*Pin*
VCC	A1	PRESENT#	B1
GND	A2	FLASH#	B2
ISPLSI#	A3	cesTRST#	B3
cesMODE	A4	cesSCLK	B4
cesSDO	A5	cesSDI	B5
SPARE_1	A6	SPARE_2	B6

NOTE

SPARE_1 and SPARE_2 are not connected.
Connector male 12p.90 Deg Type: ERNI 054594 - SMD.

Table 11-14 HP-Debug

Signal Name	*Pin*	*Signal Name*	*Pin*
TDO	1	NC	2
TDI	3	TRST#	4
NC	5	VCC3	6
TCK	7	NC	8
TMS	9	NC	10
SRESET#	11	NC	12
HRESET#	13	NC	14
CHECKSTOP#	15	GND	16

NOTE

The COP connector is optional and could be put on the bottom-side of the RIOC Pcb.
VCC3 pin 6 is used by the tools for presence detection (onboard 1 K¾ serial resistor)
2.54 mm pitch connector 2x8p. Vertical Type 3M or AMP.