Ursula PCI Manager - Functional Specification

Distribution:	COMPANY CONFIDENTIAL
Title:	Ursula PCI Manager - Functional Specification
Drawing Number:	1309,206/FS
Issue:	D ***PROVISIONAL AND SUBJECT TO CHANGE WITHOUT NOTICE***
Author(s):	W. Turner
Date:	15 Jan 1998
Change Number:	N/A
Last Issue:	N/A

Contents:

1 History

0.0.1	25 June 1997	Initial draft (released as issue A)
0.0.2	02 July 1997	Expanded section 5.
0.0.3	08 July 1997	Extended after comments on 0.0.2
0.0.4	09 July 1997	Several changes following spec review. Released as issue B.
0.0.6	15 Sept 1997	Slots are now devices, due to new information. Issue C.

2 Purpose

This document is a functional specification of the software to be placed in RISC OS for the RiscPC II (Phoebe hardware) to support the access and use of expansion cards placed on the PCI bus present in Phoebe by third party developers. The software will be referred to from here on as the 'PCI manager'. The implementation of the software is tied to the PCI9080 support chip, but the API is intended to not be hardware-specific.

3 Outstanding Issues

4 Product Overview

The aim of the software is to provide access to and use of standard PCI expansion cards under RISC OS. As delivered, the software will provide an interface to the PCI bus sufficient to allow third parties to write drivers such that PCI cards can be used in RISC OS in a similar way to how podules are utilised. In order to make life easier for the third party writers, it is proposed to use a SWI interface that is based on a subset of that used for podules.

5 Definitions and Concepts

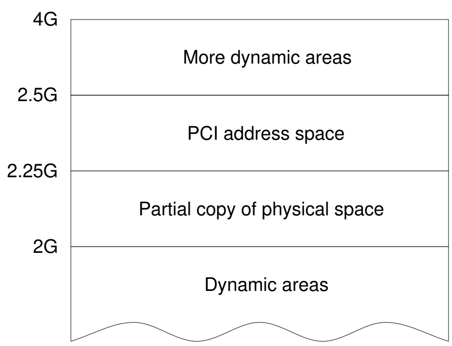
5.1 Definitions

The following is a list of terms used in this document, along with a brief description of each.

Local bus	This is a terminology used by the PCI standard, and in this case refers to the Phoebe system bus
PCI	Peripheral Component Interconnect. An expansion card system commonly used in x86 PCs.
Podule	Expansion card system used in current Acorn RiscPC systems.
Device	A PCI card can have up to 8 distinct devices on it.

5.2 Concepts

To access the cards in PCI slots, it is obviously necessary to map their devices into the Phoebe memory map at some location. For this purpose, 256MB of space has been allocated in the physical map, from 0x40000000 to 0x4FFFFFFF. In order to simplify the physical->logical mapping, 256MB of logical space has also been allocated for PCI use. This is at the top of the memory map, as shown in the diagram below, reducing the amount of space for dynamic areas by the same amount. Note that this address space will be dynamically allocated to devices on the inserted PCI cards, with the exception of 256 bytes at the 2.25G boundary, which is reserved for access to the configuration space on the PCI cards.



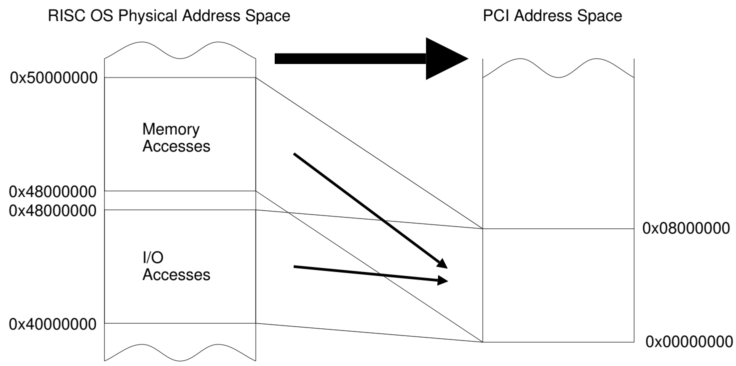
The hardware in Phoebe supports three PCI slots, accessed via standard PCI interface sockets on the NLX backplane.

Note that although only 3 slots (24 devices max) will be available on Phoebe, the software will not impose a maximum device number lower than 127, to allow for future expansion.

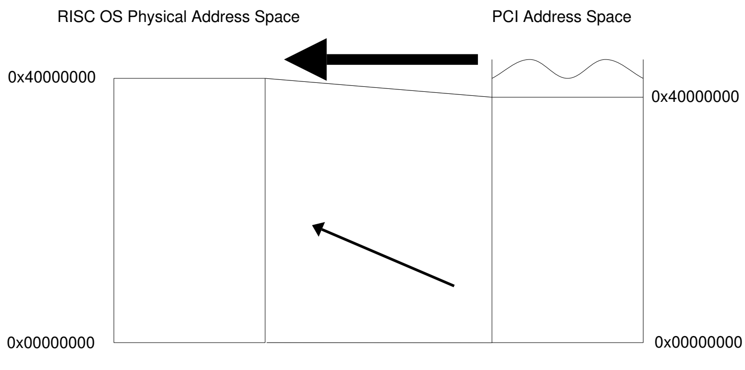
5.3 Concepts of operation

This section has an overview of the operation of the PCI support software. It mainly covers the initialisation process, but also covers how it deals with subsequent requests.

Firstly, the PCI9080 chip is programmed into its initial state, using the local bus as an EEPROM substitute. The vendor info etc (which are allocated to us by the chip manufacturer), are programmed, and the physical memory mappings for local->PCI memory access, and local->PCI I/O accesses. Note that this is a one-off boot-time mapping, and cannot be changed at runtime. The mappings set up are shown in the diagram below. Although this may appear at first to potentially allow address conflicts between memory and IO accesses from local space, this is taken care of by the controller.



Next, the PCI standard headers are interrogated on the slots, to find whether cards are present, and if so, how much memory and IO space they require for each device. These requirements are then mapped into the PCI address space, starting at address 0x00000000, so that they can be accessed from RISC OS. Next, the PCI->local mappings are set up such that they map into the local address space in a simple 1->1 direct mapping as shown below. This allows PCI cards to access the entirety of the host's memory space, with the exception of the area which is allocated to PCI. Note that this includes VidC, IOMD2 etc.



The mappings are maintained by the PCI module, and are passed onto interested parties via the PCI_AddressMapping SWI, documented in section 7.

If any expansion ROM areas were found on the PCI cards in the system, they are interrogated to see whether they contain a RISC OS image within, and if so, this is treated in the same manner as a Podule ROM.

6 User Interface

As the purpose of the software is to provide an interface for programmers, there is very little required in the way of a user interface. However, in line with the current podule support, one star commands is implemented.

*PCIDevices - Gives information on the installed PCI expansion cards

This star command is analogous to the *Podules command. However, PCI expansion cards don't hold textual information on the card to identify it, so <string> will be 'No driver. Vendor=&XXXX, Device=&XXXX', where &XXXX is a 4-digit hexadecimal number. When a driver has been found for the card, the driver may provide a replacement textual string. If the card is empty, the string will be 'No installed PCI card'.

```
PCI Expansion card 0: <string>
```

The module will also provide a help string given by *Help PCIDevices. It will report '*PCIDevices gives information about PCI expansion cards installed in the computer.'

7 Programmer Interface

In order to make the transition to PCI drivers from Podule drivers as easy as possible for third-party developers, the programmer interface is kept as close as possible to the existing Podule interface.

Service Calls

Service_PreReset

As documented in the PRMs 4-134, this call is also of use to the PCI drivers, for the same reasons. However, it doesn't require any modification for use with PCI.

Interrupt handling

The interrupt handling mechanism for PCI interrupts is aimed at being as close as possible to that currently in use for podules, given the hardware similarity. Both systems have one IRQ and one FIQ line shared between all the cards, and there will be some card-specific address that indicates an IRQ has occured. It is up to the RISC OS kernel to arbitrate as to which individual driver gets the interrupt, using the DeviceVector mechanism, as described in PRM1-116. Note that due to the changes in IOMD2, the kernel will require some work in order to support the extended range of interrupts in Phoebe.

SWI Calls

PCI_ReturnNumberOfDevices

This SWI returns the number of PCI devices in the system.

On Entry:

No entry parameters

On Exit:

R0 Number of PCI devices in the system.

This SWI returns the number of PCI devices. There can be up to 8 devices per PCI card, implying a maximum of 24 devices on Phoebe. However, the SWI should be treated as though it can return any value to allow for future expansion (such as supporting PCI bridge chips)

PCI_ReadID

This SWI returns the Vendor and ID strings from the specified PCI device. It also returns the physical slot number that the device is in (and thus the card it's on).

On Entry:

R0 Flags

All bits reserved. Should be 0.

R3 PCI device number

On Exit:

R0 PCI card information

bits	meaning
0-15	Vendor ID
16-31	Device ID

R1 PCI slot number the device is on

All other registers preserved

If there is no such PCI device, the Vendor and Device IDs are returned as 0xFFFF.

PCI_ReadHeader

This SWI reads the entire 64-byte PCI type 00h configuration space header region into the buffer pointed to by R2

On Entry:

R0 Flags

All bits reserved. Should be 0.

R2 Pointer to a buffer of 64 bytes.

R3 PCI device number

On Exit:

All registers preserved.

If there is no such PCI device, the first word of the buffer is returned as 0xFFFFFFFF, ie indicating Vendor and Device IDs of 0xFFFF. The contents of the rest of the buffer may be corrupted.

PCI_RegisterDriver

This SWI allows a module to register itself with the PCI manager. It passes the vendor/device word in, along with a bitmask (to allow for covering various devices), and it returns the device number of the first device that matches the search criteria, and does not already have a driver associated with it.

On Entry:

R0 Flags

All bits reserved. Should be 0.

R1 Vendor/Device information

bits	meaning
0-15	Vendor ID of card being looked for

16-31 Device ID of card being looked for

R2 Bitmask to indicate bits interested in in R0. A '1' bit indicates 'interested'

R4 Identifying word used in PCI_DeRegisterDriver.

R5 Pointer to textual string to use in *PCIDevices. Maximum 20 characters.

On Exit:

R3 PCI device number which represents the searched-for device. Contains &FFFF if device not found.

PCI_DeRegisterDriver

This SWI deregisters a previously registered driver.

On Entry:

R0 Flags

All bits reserved. Should be 0.

R3 PCI device number.

R4 Identifying word passed into PCI_RegisterDriver in R3.

On Exit:

R4 Zero if deregister succeeded. Preserved if deregister failed.

All other registers preserved

PCI_AddressMapping

This SWI returns the address mappings for the given device for local->PCI accesses.

On Entry:

R0 Flags

All bits reserved. Should be 0.

R1 Pointer to a word-aligned buffer to recieve the word-aligned results

R2 Length of buffer

R3 PCI device number

On Exit:

R2 Length of results

All other registers preserved

The results are returned in quad-word blocks as described below:

Word	Contents	
0	bits 0-7:	Address region on the PCI device
	bits 8-29:	Reserved
	bits 30-31:	0=Memory area, 1=IO area, 2=Expansion ROM, 3=reserved
1	PCI base address of area	
2	Local logical base address of area	
3	Size of area in bytes.	

8 Standards

The PCI hardware conforms to the PCI 2.1 standard. The software also adheres to the data structures laid down in that standard, as well as conforming to the precedent set by the RISC OS Podule standard. Also, the software has been written to allow for the implementation of the I^{2}O standard in the future, either by Acorn or a 3rd party.

9 Data Interchange and Formats

The data interchange standards and formats used are as defined in the PCI 2.1 standard

10 External Dependencies

- Phoebe prototype availability.
- Network and memory-based PCI cards.
- Phoebe physical address space (to claim the 128MB requested)
- Ursula kernel updates for IRQ/FIQ handling, DMA, Page-mapping. Logical<>Physical address conversion.

11 Acceptance Tests

Due to the tight timescales and the inherently wide range of cards available in the marketplace, it is obviously not possible to test the software under all conditions. However, the fundamentals of PCI operation will be tested independantly, namely PCI<->host transfer of memory contents, interrupt-handling, and reading of the PCI configuration space. It is suggested that the best method to achieve this is to use a standard PC video card and redirecting VDU output to the PCI framebuffer. This also has the advantage that such a system can be used to demonstrate the PCI system to third parties.

12 Product Organisation

The PCI support code will be delivered to the Ursula project as a RISC OS module. Source will be merged into the Ursula build tree. Programmer documentation for the PRMs will also be supplied.

13 Future Enhancements

Due to the tight timescales applied to the Ursula/Phoebe projects as a whole, the aim of the support software has been to provide a minimum level of support for PCI. As a result, there are a number of future enhancements possible, which are listed below:

- Support for the I^{2}O standard for PCI expansion cards with IO processors.
- Support for multi-processing over the PCI bus. This would extend to non-ARM processors (eg PPC/x86)
- Support for PCI bridge chips in the 2 on-board sockets.

14 References

PCI Specification, revision 2.1. Published by the PCI Special Interest Group, 1995.

PCI 9080 PCI Busmaster chip technical documentation. Published by PLX Technology.

Digital Semiconductor 21143 PCI/Cardbus 10/100-Mb/s Ethernet LAN controller hardware reference manual.

I2O SIG, http://www.i2osig.org