OPENPCI 2

The openpci.library is a library who support all kind of PCI busboard on Amiga and PC (with Amithlon through powerpci.library). This library as done for be a standard on all future PCI/AGP driver in the Amiga market.

TABLE OF CONTENTS

openpci.library/COPYRIGHT

COPYRIGHT NOTICE

Thanks to Ralph Schmidt, MorphOS Team http://www.morphos.net,
http://www.pegasosppc.com,
Vision Factory Development (VFD), Frank Mariak, Robert Reiswig http://www.vgr.com,
Matay (Filip Dab-Mirowski) http://www.matay.pl, **Harald Frank** http://www.vmc.de,
Nicolas Sallin (Henes), Antoine Dubourg (Tcheko), Alkis Tsapanidis (AmiGR),
IO moral support and all other Amiga users in the world.
Special thanks to Taniusha for moral support.

Official OpenPci project and driver WWW page :
http://bvernoux.free.fr/DevPCI.php

DISCLAIMER

openpci.library/--changes--

29/02/2008
- Added OpenPci2.odt open document.
- Added vbcc specific inline and libraries (thanks to Christoph Fassbach).
- fixed libraries/openpci.h (thanks to Christoph Fassbach).

24/04/2006
- Thanks to Stefan A. Haubenthal about include problems for C++.
- Updated documentations and includes (Include/libraries/openpci.h, Include/clib/openpci_protos.h, Include/gcc/inline/openpci.h, Include/fd/openpci_lib.fd, doc/OpenPci2.txt, OpenPci2.rtf, OpenPci2.pdf for compatibility with C++ (renamed class to devclass).
- Removed OpenPci2.doc.

06/03/2004
- pci_physic_to_logic_addr() function added
- pci_obtain_card() function added
- pci_release_card() function added
- pci_logic_to_physic_addr()/pci_physic_to_logic_addr() documentation clarified in fact we can use these functions for any type of memory (DMA memory, PCI card memory …)
- Logical/Virtual Address and Physical Address paragraph updated
- Documentation cleanup/typo fix
- Updated libraries/openpci.h updated now use the define “MIN_OPENPCI_VERSION” when you open the openpci.library
- DriverExample.c updated now it use pci_obtain_card()/pci_release_card() and pci_physic_to_logic_addr()

25/05/2003
- pci_bus() documentation correction
- pci_find_slot() documentation updated
- pci_add_intserver() removed some errors in doc : “The LN_NAME of the interrupt structure must point to a NULL value because it's an OpenPCI reserved field.” This things is wrong in fact you can fill this field
- C source example : DriverExample dir added
- Some cleanup in the documentation

08/04/2003
- release codes: C example
- pci_read_conf_long : more information, fix
- new chapter : --Developer Informations-- added
- new flags : MEM_24BITDMA for pci_allocdma_mem()

21/02/2003
- pci_add_intserver : fix
- new libraries/openpci.h

openpci.library/--background--

Purpose

The openpci.library is a library supporting all kind of PCI busboards on Amiga (Grex, Prometheus..etc), PC (with Amithlon, through powerpci.library) and Pegasos1/2. The aim of this library is to be a standard for all future PCI/AGP drivers in the Amiga/MorphOS market.

Contents

The structures you have to use are described in the openpci.h include file for the c/c++ programmers.

To use this library, it is very usefull to have the proper pci2.1 or pci2.2 documentation handy. It is available from the pcisig at http://www.pcisig.org

openpci.library/pci_bus

NAME

pci_bus - Try to find which PCI bus is available in the computer

SYNOPSIS

```
USHORT bus = pci_bus()
             D0
```

OFFSET

-30

FUNCTION

Return a value corresponding to a PCI bus (see libraries/openpci.h) or 0 if no PCI bus are detected.

RESULTS

Possible Value :

```
#define MediatorA1200Bus 0x1
#define MediatorZ4Bus    0x2
#define PrometheusBus    0x4
#define GrexA1200Bus     0x8
#define GrexA4000Bus     0x10
#define PegasosBus       0x20
#define PowerPciBus      0x40
```

EXCEPTIONS

SEE ALSO

BUGS

openpci.library/pci_inb

NAME
pci_inb - Read a byte on a PCI board at the given address

SYNOPSIS
UCHAR value = pci_inb(ULONG address)
D0 A0

OFFSET
-36

FUNCTION
Return the value read from the address.

INPUTS
ULONG address - Address of I/O or Memory PCI board.

RESULTS
UCHAR value - Value read on the board.

EXCEPTIONS

SEE ALSO
pci_outb()

BUGS

openpci.library/pci_outb

NAME
pci_outb - Write a byte (8bits) to a PCI board at the given address

SYNOPSIS
VOID pci_outb(UCHAR value, ULONG address)
D0 A0

OFFSET
-42

FUNCTION
Write the value to the address

INPUTS
UCHAR value - Value to write in the address

ULONG address - Address where we write the value

RESULTS
None

EXCEPTIONS

SEE ALSO
pci_inb()

BUGS

openpci.library/pci_inw

NAME
pci_inw - Read a word (16bits) on a PCI board at the given address

SYNOPSIS
USHORT value = pci_inw(ULONG address)
D0 A0

OFFSET
-48

FUNCTION
Return the value read from the address.

INPUTS
ULONG address - Address of I/O or Memory PCI Board.

RESULTS
USHORT value - Value read on the board.

EXCEPTIONS

SEE ALSO
pci_outw()

BUGS

openpci.library/pci_outw

NAME
pci_outw - Write a word (16bits) to a PCI board at the given address given

SYNOPSIS
VOID pci_outw(USHORT value, ULONG address)
D0 A0

OFFSET
-54

FUNCTION
Write the value to the address

INPUTS
USHORT value - Value to write in the address

ULONG address - Address where we write the value

RESULTS
None

EXCEPTIONS

SEE ALSO
pci_inw()

BUGS

openpci.library/pci_inl

NAME
pci_inl - Read a long (32bits) on a PCI board at the given address

SYNOPSIS
ULONG value = pci_inl(ULONG address)
D0 A0

OFFSET
-60

FUNCTION
Return the value read from the address.

INPUTS
ULONG address - Address of I/O or Memory PCI Board.

RESULTS
ULONG value - Value read on the board.

EXCEPTIONS

SEE ALSO
pci_outl()

BUGS

openpci.library/pci_outl

NAME
pci_outl - Write a long (32bits) to a PCI board at the given address

SYNOPSIS
VOID pci_outl(ULONG value, ULONG address)
D0 A0

OFFSET
-66

FUNCTION
Write the value to the address

INPUTS
ULONG value - Value to write in the address

ULONG address - Address where we write the value

RESULTS
None

EXCEPTIONS

SEE ALSO
pci_inl()

BUGS

openpci.library/pci_to_hostcpy

NAME
pci_to_hostcpy - Copy a memory block from DMA Memory to Amiga Host Memory, with size of packetsize

SYNOPSIS
VOID pci_to_hostcpy(VOID *pcimemsrc,VOID *memdest,ULONG packetsize)
A0 A1 D0

OFFSET
-72

FUNCTION
Write pcimemsrc buffer (Memory on a PCI card) with size of packetsize to memdest buffer (Memory on the Host)

INPUTS
VOID *pcimemsrc - Memory source (DMA Memory) - must be 8 bytes aligned

VOID *memdest - Memory destination (Memory address on Amiga Host) - must be 8 bytes aligned

ULONG packetsize - Size in bytes of the memory to be copied

RESULTS
None

EXCEPTIONS

SEE ALSO
host_to_pcicpy(), pci_to_pcicpy()

BUGS

openpci.library/host_to_pcicpy

NAME

host_to_pcicpy - Copy a memory block from Amiga Host Memory to DMA Memory with size of packetsize

SYNOPSIS

```
VOID host_to_pcicpy(VOID *memsrc,VOID *pcimemdest,ULONG packetsize)
                                    A0                A1                D0
```

OFFSET

-78

FUNCTION

Write memsrc buffer (Memory on Amiga Host) with size of packetsize to memdest buffer (DMA Memory)

INPUTS

VOID *memsrc - Memory source (Memory address on Amiga Host) - must be 8 bytes aligned

VOID *pcimemdest - Memory destination (DMA Memory) - must be 8 bytes aligned

ULONG packetsize - Size in bytes of the memory to be copied

RESULTS

None

EXCEPTIONS

SEE ALSO

pci_to_hostcpy(), pci_to_pcicpy()

BUGS

openpci.library/pci_to_pcicpy

NAME
pci_to_pcicpy - Copy a memory block from DMA Memory to DMA Memory

SYNOPSIS
VOID pci_to_pcicpy(VOID *pcimemsrc, VOID *pcimemdest, ULONG packetsize)
A0 A1 D0

OFFSET
-84

FUNCTION
Copy from DMA Memory to DMA Memory with size of packetsize

INPUTS
VOID *pcimemsrc - Memory source (DMA Memory) - must be 8 bytes aligned

VOID *pcimemdest - Memory destination (DMA Memory) - must be 8 bytes aligned

ULONG packetsize - Size in bytes of the memory to be copied

RESULTS
None

EXCEPTIONS

SEE ALSO
pci_to_hostcpy(), host_to_pcicpy()

BUGS

openpci.library/pci_find_device

NAME
pci_find_device - Try to find the specified PCI device on the PCI bus.

SYNOPSIS
struct pci_dev *pcidev = pci_find_device(USHORT vendor, USHORT device, struct pci_dev *pcidev)
D0 D0 D1 A0

OFFSET
-90

FUNCTION
Find the first device that matches the given vendor/device specifications.

To start the search from the beginning of the PCI devices list, please use NULL as the pcidev structure pointer.

NOTE
Use ULONG 0xFFFF as a wildcard for any entry.

INPUTS
UWORD vendor - From pcisig assigned vendor ID

UWORD device - Manufacturer assigned product ID

struct pci_dev *pcidev - pci_dev structure pointer to start research on one PCI bus or on all PCI bus if pcidev=NULL (0)

RESULTS
struct pci_dev *pcidev - valid pcidev pointer if success or NULL on failure.

EXCEPTIONS

SEE ALSO
pci_find_class(), pci_find_slot()

BUGS

openpci.library/pci_find_class

NAME
pci_find_class - Try to find the specified class on the PCI bus.

SYNOPSIS
struct pci_dev *pcidev = pci_find_class(ULONG devclass, struct pci_dev *pcidev)
D0 D0 A0

OFFSET
-96

FUNCTION
Find specified Class ID, starting at pcidev structure or search on all PCI bus if pcidev=NULL (0)

INPUTS
ULONG devclass - Class ID to search (for more information see PCI2.1 spec).

struct pci_dev *pcidev - pci_dev structure pointer to start search on
or search on all PCI bus if pcidev=NULL (0)

RESULTS
struct pci_dev *pcidev - valid pcidev pointer if success or NULL on failure.

EXCEPTIONS

SEE ALSO
pci_find_device(), pci_find_slot()

BUGS

openpci.library/pci_find_slot

NAME
pci_find_slot - Try to find the specified PCI device on the specified bus.

SYNOPSIS
struct pci_dev *pcidev = pci_find_slot(UCHAR bus, ULONG devfn)
D0 D0 D1

OFFSET
-102

FUNCTION
Find the specified slot and function encoded in devfn on the specified PCI bus,
if bus=0 search on all PCI bus.
(default first PCI bus start at 1).

INPUTS
UBYTE bus - the PCI bus number to look on

ULONG devfn - the device number (called slot too) and the device function (func) number to look for
(see libraries/openpci.h for PCI_DEVFN(), PCI_SLOT(), PCI_FUNC() macro conversion function)

RESULTS
struct pci_dev *pcidev - valid pci_dev structure pointer if success or NULL (0) on failure.

EXCEPTIONS

SEE ALSO
pci_find_device(), pci_find_class()

BUGS

openpci.library/pci_read_config_byte

NAME
pci_read_config_byte - Read a byte (8bits) from configuration space of a device

SYNOPSIS
UBYTE config = pci_read_config_byte(UBYTE registernum, struct pci_dev *pcidev)
D0 D0 A0

OFFSET
-108

FUNCTION
Read a byte in the pcidev specified and at the registernum in the PCI config space.

INPUTS
UBYTE registernum - The register num addr to start reading in PCI config space

struct pci_dev *pcidev - A pci_dev structure pointer where you want to read PCIconfig space.

RESULTS
UBYTE config - Return the value read in the config space for the specified registernum offset
and the pci_dev struct pointer.

EXCEPTIONS

SEE ALSO
pci_write_config_byte()

BUGS

openpci.library/pci_read_config_word

NAME
pci_read_config_word - Read a word (16bits) from configuration space of a device

SYNOPSIS
USHORT config = pci_read_config_word(UBYTE registernum, struct pci_dev *pcidev)
D0 D0 A0
OFFSET
-114

FUNCTION
Read a word in the pcidev specified and at the registernum in the PCI config space.

INPUTS
UBYTE registernum - The register num addr to start reading in PCI config space

struct pci_dev *pcidev - A pci_dev structure pointer where you want to read PCI config space.

RESULTS
USHORT config - Return the value read in the config space for the specified registernum offset
and the pci_dev structure pointer.

EXCEPTIONS

SEE ALSO
pci_write_config_word()

BUGS

openpci.library/pci_read_config_long

NAME
pci_read_config_long - Read a long (32bits) from configuration space of a device.

SYNOPSIS
ULONG config = pci_read_config_long(UBYTE registernum, struct pci_dev *pcidev)
D0 D0 A0

OFFSET
-120

FUNCTION
Read a long in the pcidev specified and at the registernum in the PCI config space.

INPUTS
UBYTE registernum - The register num addr to start reading in PCI config space.

struct pci_dev *pcidev - A pci_dev structure pointer where you want to read PCI config space.

RESULTS
ULONG config - Return the value read in the config space for the specified registernum offset
and the pci_dev structure pointer.

EXCEPTIONS

SEE ALSO
pci_write_config_long()

BUGS

openpci.library/pci_write_config_byte

NAME
pci_write_config_byte - Write a byte (8bits) in configuration space of a device.

SYNOPSIS
VOID pci_write_config_byte(UBYTE registernum, UBYTE val, struct pci_dev *pcidev)
D0 D1 A0

OFFSET
-126

FUNCTION
Write a byte with the value of val in the PCI config space with in the pcidev specified and the registernum.

INPUTS
UBYTE registernum - The register num addr to start writing in PCI config space.

UBYTE val - The value to store in PCI config space.

struct pci_dev *pcidev -- A pci_dev structure pointer where you want to read PCI config space.

RESULTS
None

EXCEPTIONS

SEE ALSO
pci_read_config_byte()

BUGS

openpci.library/pci_write_config_word

NAME
pci_write_config_word - Write a word (16bits) in configuration space of a device.

SYNOPSIS
VOID pci_write_config_word(UBYTE registernum, USHORT val, struct pci_dev *pcidev)
D0 D1 A0

OFFSET
-132

FUNCTION
Write a word with the value of val in the PCI config space with in the pcidev specified and the registernum.

INPUTS
UBYTE registernum - The register num addr to start writing in PCI config space.

USHORT val - The value to store in PCI config space.

struct pci_dev *pcidev - A pci_dev structure pointer where you want to read PCI config space.

RESULTS
None

EXCEPTIONS

SEE ALSO
pci_read_config_word()

BUGS

openpci.library/pci_write_config_long

NAME

pci_write_config_long - Write a long (32bits) in configuration space of a device.

SYNOPSIS

VOID pci_write_config_long(UBYTE registernum, ULONG val, struct pci_dev *pcidev)
D0 D1 A0

OFFSET

-138

FUNCTION

Write a long with the value of val in the PCI config space with in the pcidev specified.
and the registernum.

INPUTS

UBYTE registernum - The register num addr to start writing in PCI config space.

ULONG val - The value to store in PCI config space.

struct pci_dev *pcidev - A pci_dev structure pointer where you want to read PCI config space.

RESULTS

None

EXCEPTIONS

SEE ALSO

pci_read_config_long()

BUGS

openpci.library/pci_set_master

NAME
pci_set_master - Set the specified device in Bus Master mode.

SYNOPSIS
BOOL result = pci_set_master(struct pci_dev *pcidev)
D0 A0

OFFSET
-144

FUNCTION
Test if the device is Bus Master capable and if it's capable set the device to Bus Master.

INPUTS
struct pci_dev *pcidev - A pci_dev structure pointer where you want to set the Bus Master.

RESULTS
Return TRUE if success or FALSE if failure.

EXCEPTIONS

SEE ALSO

BUGS

openpci.library/pci_add_intserver

NAME

pci_add_intserver - Add an interrupt server to a system server chain.

SYNOPSIS

BOOL result = pci_add_intserver(struct Interrupt *PciInterrupt, struct pci_dev *pcidev)
D0 A0 A1

OFFSET

-150

FUNCTION

This function adds a new interrupt server to a pci_dev structure server chain.
The node is located on the chain in a priority dependent position.

Each link in the chain will be called in priority order until the
chain ends or one of the servers returns with the 68000's Z condition
code clear (indicating non-zero). Servers on the chain should return
with the Z flag clear if the interrupt was specifically for that
server, and no one else. VERTB servers should always return Z set.
(Take care with High Level Language servers, the language may not
have a mechanism for reliably setting the Z flag on exit).

Servers are called with the following register conventions:

D0 - scratch
D1 - scratch

A0 - scratch
A1 - server is_Data pointer (scratch)

A5 - jump vector register (scratch)
A6 - scratch

All other registers must be preserved

NOTE

To receive interrupts you MUST manualy enable inside your hardware
the requested interrupt sources.

INPUTS

struct pci_dev *pcidev - A pci_dev structure pointer where you want to set the interrupt server.

struct Interrupt *PciInterrupt - pointer to an Interrupt structure.

RESULTS

Return TRUE if success or FALSE if failure.

EXCEPTIONS

SEE ALSO

pci_rem_intserver()

BUGS

openpci.library/pci_rem_intserver

NAME
pci_rem_intserver - Remove an interrupt server from a pcidev server chain.

SYNOPSIS
VOID pci_rem_intserver(struct Interrupt *PciInterrupt, struct pci_dev *pcidev)
A0 A1

OFFSET
-156

FUNCTION
This function removes an interrupt server node from the given pcidev server chain.

INPUTS
struct pci_dev *pcidev - A pci_dev structure pointer where you want to remove the interrupt server.

struct Interrupt *PciInterrupt - pointer to the Interrupt structure used before by pci_add_intserver().
Don't modify any field in this structure.

RESULTS
None

EXCEPTIONS

SEE ALSO
pci_add_intserver()

BUGS

openpci.library/pci_allocdma_mem

NAME

pci_allocdma_mem - Allocate DMA memory, Logical/Virtual address seen by the CPU for a specified PCI device.

SYNOPSIS

```
APTR buffer = pci_allocdma_mem(ULONG size, ULONG flags)
        D0                                  D0          D1
```

OFFSET

-162

FUNCTION

Allocate memory for DMA usage depending on the flags used.
(see libraries/openpci.h for the flags)
return Logical/Virtual address seen by the CPU.

INPUTS

ULONG size - Size of memory to allocate, must be divisible by 8.

ULONG flags - Type of memory to allocate:
MEM_PCI For graphics board memory
MEM_NONCACHEABLE For fast NonCacheable memory
MEM_24BITDMA For memory in the lower 24bits memory area

RESULTS

Return memory pointer on success or NULL if failure.
The address returned is a Logical/Virtual address seen by the CPU.

EXCEPTIONS

SEE ALSO

pci_freedma_mem()

BUGS

openpci.library/pci_freedma_mem

NAME
pci_freedma_mem - Free DMA memory of a specified PCI device.

SYNOPSIS
VOID pci_freedma_mem(APTR buffer,ULONG size)
A0 D0

OFFSET
-168

FUNCTION
Free DMA memory previously allocated by pci_allocdma_mem().

INPUTS
APTR buffer - Buffer (Logical/Virtual address seen by the CPU)
previously returned by pci_allocdma_mem().

ULONG size - Size of the previously allocated memory.

RESULTS
None

EXCEPTIONS

SEE ALSO
pci_allocdma_mem()

BUGS

openpci.library/pci_logic_to_physic_addr

NAME
pci_logic_to_physic_addr - Convert Logical/Virtual address seen by the CPU to Physical address
seen by the PCI bus.

SYNOPSIS
APTR PciPhysicalAddr = pci_logic_to_physic_addr(APTR PciLogicalAddr, struct pci_dev *pcidev)
D0 A0 A1

OFFSET
-174

FUNCTION
Convert Logical/Virtual address seen by the CPU to Physical address seen by the PCI bus.

INPUTS
APTR PciLogicalAddr - PCI Logical/Virtual address seen by the CPU.
(for example address returned by pci_allocdma_mem())

struct pci_dev *pcidev - A pci_dev structure pointer where you want to convert the Logical/Virtual to Physical address.

RESULTS
APTR PciPhysicalAddr - PCI Physical address seen by the PCI bus or NULL if error.

EXCEPTIONS

SEE ALSO
pci_allocdma_mem()

BUGS

openpci.library/pci_physic_to_logic_addr

NAME
pci_physic_to_logic_addr - Convert Physical address seen by the PCI bus to Logical/Virtual address
seen by the CPU.

SYNOPSIS
APTR PciLogicalAddr = pci_physic_to_logic_addr(APTR PciPhysicalAddr, struct pci_dev *pcidev)
D0 A0 A1

OFFSET
-180

FUNCTION
Convert Physical address seen by PCI bus to Logical/Virtual address seen by the CPU.

INPUTS
APTR PciPhysicalAddr - PCI Physical address seen by the PCI bus.

struct pci_dev *pcidev - A pci_dev structure pointer where you want to convert the physical address to logical/virtual address.

RESULTS
APTR PciLogicalAddr - PCI Logical/Virtual address seen by the CPU or NULL if error.

EXCEPTIONS

SEE ALSO
pci_logic_to_physic_addr()

BUGS

openpci.library/pci_obtain_card

NAME
pci_obtain_card - Try to do a software lock of a PCI card.

SYNOPSIS
BOOL result = pci_obtain_card(struct pci_dev *pcidev)
D0 A0

OFFSET
-186

FUNCTION
Try to lock a PCI card specified by pcidev structure.

INPUTS
struct pci_dev *pcidev - A pci_dev structure pointer where you want to lock a PCI card.

RESULTS
Return TRUE if success or FALSE if failure.

EXCEPTIONS

SEE ALSO
pci_release_card()

BUGS

openpci.library/pci_release_card

NAME
pci_release_card - Try to do a software unlock of a PCI card.

SYNOPSIS
VOID = pci_release_card(struct pci_dev *pcidev)
A0

OFFSET
-192

FUNCTION
Try to unlock a previously locked PCI card specified by pcidev structure.

INPUTS
struct pci_dev *pcidev - A pci_dev structure pointer where you want to unlock a PCI card.

RESULTS
None

EXCEPTIONS

SEE ALSO
pci_obtain_card()

BUGS

openpci.library/--Developer Informations--

WARNING pcidev structure !!!

The pcidev structure returned by pci_find_slot()/pci_find_device()/pci_find_slot() is public
but is READ ONLY you must NEVER write in this structure (see libraries/openpci.h).

WARNING PCI read/write !!!

When you read or write things in PCI I/O, Mem or DMA memory think ALWAYS the CPU side is in Big Endian format.
Even if it's not the case (like for X86 CPU).
It's the only way to have portable driver under any type of CPU.
For that use ALWAYS swap macro of the MorphOS Team included by libraries/openpci.h.
This macro are :
SWAPWORD(unsigned short value) : return swaped 16bits value (Big Endian <-> Little Endian conversion)
SWAPLONG(unsigned long value) : return swaped 32bits value (Big Endian <-> Little Endian conversion)

- If the PCI/AGP card you work on use Big Endian byte format redefine SWAPWORD and SWAPLONG after #include <lirbaries/openpci.h> like that :
#undef SWAPLONG
#undef SWAPWORD
#define SWAPLONG(x) (x)
#define SWAPWORD(x) (x)

When you make a driver use ALWAYS this function : pci_inb/w/l()/pci_outb/w/l()
- NEVER use direct hardware access for read/write in I/O, Mem addr of the PCI card.
It's the same rules for pci_to_hostcpy(),host_to_pcicpy(),pci_to_pcicpy():
You want to copy a memory block from Host memory to DMA memory, I/O or Mem space use host_to_pcicpy().
You want to copy a memory block from PCI DMA memory, I/O or Mem space use pci_to_hostcpy().
You want to copy a memory block from PCI DMA memory, I/O or Mem space to DMA memory, I/O or Mem space use pci_to_pcicpy().

This things are VERY important else your driver will don't work on all PCI bus supported by OpenPCI API
and with this things future OpenPCI library version will always works with your driver.

I/O, Mem or ROM base address

There's two way for found I/O, Mem or ROM base address for a specific PCI card.
At first time you must retrieve the pcidev structure with pci_find_slot()/pci_find_device()/pci_find_slot()

1) First solution (the easiest)
The pcidev structure contains base_address from 0 to 5 included who is the I/O or Mem Base address
of each PCI card detected (see OpenPciInfo.c source for more information on how to detect
if it's an I/O or a Mem address...etc).
For I/O,Mem base address and base size use pcidev structure base_address[] for base address,
base_size[] for the size of I/O or Mem used.
For ROM base use pcidev rom_address for the rom base address and rom_size for the rom size.
Look OpenPciInfo.c source code for more information.

2) Second solution (require good PCI bus knowledge)

Use pci_read_config_long() with the flags PCI_BASE_ADDRESS_X=registernum for retrieve the base address of
a specified pcidev.
Example :
pci_write_config_long(PCI_BASE_ADDRESS_0,0xFFFFFFFF,pcidev)
base_size0=pci_read_config_long(PCI_BASE_ADDRESS_0,pcidev)
/* return Size with format specified by pci2.x spec */
base_addr0=pci_read_config_long(PCI_BASE_ADDRESS_0,pcidev)
/* second read return BaseAddr0 */

Read PCI 2.x specification for more information.

Logical/Virtual Address and Physical Address

When you use pci_read_config_long(PCI_BASE_ADDRESS_X,pcidev) this call will return ALWAYS the Logical/Virtual Address seen by CPU, use pci_logic_to_physic_addr() to convert this address to Physical Address seen by PCI bus.

If you retrieve a physical address seen by PCI bus you can convert this address to Logical/Virtual address
with pci_physic_to_logic_addr().

The more clean way is to use the pcidev structure base_address[] (libraries/openpci.h)

How to Enable I/O, Memory and BusMastering on a PCI Card ?

First you must detect the card with pci_find_slot()/pci_find_device()/pci_find_slot(),
after you just need to enable the I/O and Mem address see the C source example in section –release_codes--.
By default on some computer the I/O and Mem are not enabled at boot (by the BIOS/OF)
it's for that you need to force this on all driver.

openpci.library/--source_codes--

C source examples:

- See DriverExample.c source (in DriverExample dir)

This PPC MOS and 68k example show how to :
- Find a PCI card on the bus (with his vendorid and deviceid).
- Enable PCI IO, Mem address space and enable Busmaster on a PCI card.
- Do DMA memory allocation and logical DMA memory -> physical DMA memory conversion.
- Do Physical DMA memory alignement.
- Do pci_add_intserver()/pci_rem_intserver() with MOS Gates.

- See OpenPciInfo.c source (in OpenPciInfo dir).