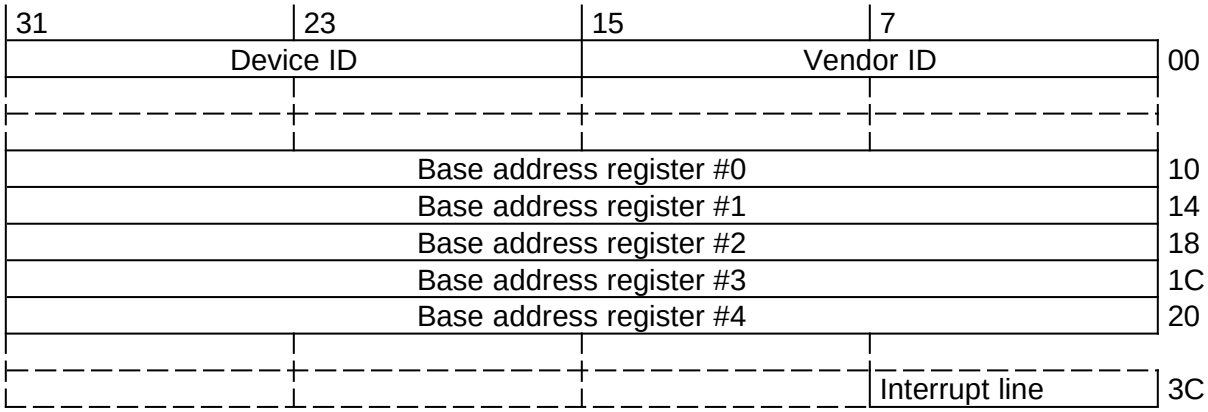
Introduction

The PCI bus offers simple “plug and play” capability to the end user in addition to other major advantages over the ISA bus. Users benefit because there are no switches or jumpers to be set in selecting the base address and interrupt required by a PCI data acquisition board like the PCI-DAS1602/16. The PCI BIOS makes these assignments when the system boots thereby guaranteeing that the your software can access the board. Contrast this with the ISA bus, where either a misplaced base address switch or a base address that conflicted with other hardware meant that one could not access the board in question.

How to determine the base address and interrupt of a PCI board

Since the base addresses of various registers in a board are set by the PCI BIOS, you will need to query the BIOS to determine these. This must be done before you access and/or program the board in any way. Before delving into the details we need to look at the PCI Configuration Header Space.

Each PCI bus device contains a unique 256-byte region called its configuration header space. The organization of this space is given in the table below. The important fields that are relevant to this discussion are named; the others are either reserved or not used.



The device and vendor IDs are unique values that are we assign to each PCI data acquisition board. These are used in querying the PCI BIOS for a specific board. The base address registers provide a mechanism for assigning I/O space. Registers in ComputerBoards’ family of PCI boards are offset from the I/O address contained in a specific base address register. For example, in the PCI-DAS1602/16 the ADC data can be read at the I/O address contained in base address register 2 and the DAC can be updated by writing to the I/O address contained in base address register 4.

The following code snippets describe the steps necessary to determine the contents of the various base address registers. The assembly language code is for Windows 3.1 or DOS. For the sake of clarity and brevity error handling code is not shown.

```
; ;PCI function values
PCI_FUNC_ID         equ 0b1h
FIND_PCI_DEV        equ 02h
READ_CONFIG_BYTE    equ 08h
READ_CONFIG_WORD    equ 09h
READ_CONFIG_DWORD   equ 0ah

;Configuration register offsets
VID_ADDR    equ 00h
DID_ADDR    equ 02h
BADR0_ADDR  equ 10h
BADR1_ADDR  equ 14h
BADR2_ADDR  equ 18h
BADR3_ADDR  equ 1ch
BADR4_ADDR  equ 20h
INTLN_ADDR  equ 3ch

; Operation registers offsets
INTCSR_ADDR equ 38h
BMCSR_ADDR  equ 3ch

;Misc. equates
IO_ADDR_MASK    equ 0fffch
INTCSR_DWORD    equ 00FF1F00H ; dword written to INTCSR_ADDR
BMCSR_DWORD     equ 08000000H ; dword written to BMCSR_ADDR

;*****************************************************************
; Macro to write a 32-bit dword to an I/O port. This is called
; from 16-bit code segment.
; Arguments : dwValue = DWORD value to write to port
; Uses      : dx must have the port address
; Returns   ; eax=dwValue
; Destroys  : eax
;----------------------------------------------------------------
OUTD MACRO dwValue
 db 66h         ; OPSIZE:
 db 0b8h        ; mov eax
 dd dwValue     ; OPSIZE : mov eax, dwValue
 db 66h, 0efh   ; OPSIZE : out dx, eax
ENDM

;STEP 1 : Find the specific board.
mov    dx, VENDOR_ID                ; vendor id available from ComputerBoards
mov    cx, PCI_DAS1602_DID      ; device id available from ComputerBoards
mov    si, 0
mov    ah, PCI_FUNC_ID     ;load pci function id into AH
mov    al, FIND_PCI_DEV    ;load search for device function into AL
```

```
int     1ah                     ;call bios ; Output:   bl = device & function number
                                ;                      bh = bus number
                                ;                      CF set if error

;STEP 2 : Read the base address registers, using BX obtained earlier
mov     di, BADR0_ADDR      ;read base addr0
mov     ah, PCI_FUNC_ID     ;load PCI function id
mov     al, READ_CONFIG_DWORD   ;load read double word function
int     1ah                 ;call bios ; Output:   ecx= register value, CF set on error
and     cx,  IO_ADDR_MASK
;At this point CX has the address contained in base address register 0. Repeat the
above five instructions with the offsets of the other base address register in DI. You
can store the contents of CX in some variable to avoid having to access the BIOS
repeatedly.

; STEP 3 Find the interrupt used by the board.
mov     di,INTLN_ADDR   ;read the interrupt line reg
mov     ah, PCI_FUNC_ID     ;load PCI function id
mov     al, READ_CONFIG_BYTE   ;load read byte function
int     1ah             ;bios interrupt call returns ch is zero, cl is IRQ#

;STEP 4 : Reset and enable interrupts if the board uses interrupts.
; First, copy base address 0 obtained earlier into DX
add     dx, BMCSR_ADDR
OUTD    BMCSR_DWORD         ;reset mail box interrupts
sub     dx, 4               ;dx=INTCSR_ADDR
OUTD    INTCSR_DWORD        ;reset pending interrupts and enable interrupts
```

The OUTD macro allows 32-bit I/O write to the Interrupt Control/Status register and Bus Master Control/Status Register.