小華的部落格

將自己踏入BIOS領域中所學習到的知識做一些心得整理，像是Legacy BIOS、EFI BIOS、Windows Driver...etc. ※版權與智慧財產權聲明:保留所有法律權利。我在寫文章時如果有引用到其他人的地方我會盡量說明參考出處，如果有遺漏的地方請告訴我，我會馬上註明! 而轉貼我的文章時也請您註明出處!

星期三, 10月 10, 2007

[我所知道的BIOS]->[PCI SCAN] 9

這次要提的是: PCI !

[About PCI device]
1. 每一個PCI device都有其 unique PFA(PCI Function Address). PFA由 bus number,device number & function number所組成.

Ex. USB device PFA is (0,6,0) <- USB is a PCI device and its bus/dev/function is 0/6/0

2. 有了PFA,就可以存取其 PCI configuration registers.

Ex. write USB PCI register 43h bit1 = 1
=>
mov eax, 80003040h
mov dx, 0cf8h
out dx, eax

```
mov dx, 0cffh
in al, dx
or al, 00000010b
out dx, al
```

* IO port 0cf8/0cfc 為 PCI config address port & data port,意即:將 address(80003040h)送到config port(0cf8h), 然後從 data port(0cfch + 3)來存取 data(al)

* 注意: 32-bit address(80003040h) 中 bit[1:0] = 00b(固定的),所以雖然存取的是 43h,但還是寫成40h ! 而要存取到 43h,則從 0cfch+3來達成 (因為: 0cfch<-> 40h,0cfdh<->41h,0cfeh<->42h, 0cffh<->43h)

3. 基本的PCI device的 config registers可分成 2 parts:

A. header region(offset 00h~3Fh)
B. device specific region(40h~FFh)

在BIOS's PCI_SCAN stage中,會touch到 part A. Ex. command byte, BARs, Interrupt line, latency timer,...etc. 而Part B是製作 or design這個device的廠商所附加的 function/feature.

4. 每個PCI device都可以 request 之前所提的 4 resources:
A. memory resource:透過 Base Address Register(BAR)
B. IO resource:透過 Base Address Register(BAR)
C. Interrupt: 透過 interrupt pin
D. DMA: 這需要 device本身即具有 bus master function(status byte會indicate)

[Why need PCI SCAN]
現在的computer system泰半由許多PCI devices所組成,因此,BIOS POST中另一個重要的 task is : PCI_SCAN !!!

它代表的是: BIOS會掃瞄 whole system,找出所有的PCI devices; initial them and build a linked list of PCI devices.在此list中的每一個node都代表一個PCI device,且含有其 characteristics !

Ex. Vendor ID,Device ID, PFA,Option ROM exist or NOT,...etc.

一旦建好此表,以後的 tasks 隨時都可以參考 !!!

所以, after PCI_SCAN,有兩件事完成了:
1. PCI device initialization;device config registers(Part A) are correctly set ...

2. One data structure is built to describe the PCI devices in whole system(建在memory中)

這也是屬於kernel code part ^_^ (system 一般很少 hang at this stage...)

符合PCI spec的device即稱為.....PCI device ^_^

[補充] PCIe device

PCIe device => 符合PCIe spec的device(...廢話...)

對軟體而言,它仍是PCI device. 因此,基本的 header region and device specific region也有. 不過,PCIe新定義了extended config space,即 offset 100h(含)以上,直到 FFFh(所以, 最大可以至 4096 bytes)

存取 PCI config space的方式,用原來 0cf8/0cfc的方式依然可行,但只能 access offset 00h~FFh. 要 access 100h(含)以上的 extended config space,則必須用 memory transaction的方式 !

Ex. mov ax, [50400000h] <- read device (4,0,0)'s register 0;2 bytes

here 50000000h: PCIe extended base address. 可以從 chipset register得知
bit[27:20]: Bus information
[19:15]: Device information
[14:12]: Function information
[11: 8]: Extended Register
[7:2]: DW number
[1:0]: Byte enable

因此,只要知道 PCIe extended base address,就可以像以前一樣,可以任意存取 PCIe config registers, even > 0FFh !

除此之外, PCIe device可以由其 Capability pointer(points to a linked list of capabilities)辨認出來. 因為,在眾多的capabilities中,會有一個 PCIe capability;其 ID value = 10h.

Note: PCIe extended base address 要 reserve and report to OS. Size is 256MByte. 這是BIOS需要做的. (當然,BIOS也要將此 base address寫入 chipset register,讓 chipset 知道:有這樣的 cycle時,是給PCIe device的 !)

[補充] For P2P bridge

P2P bridge = PCI-to-PCI bridge. 其存在可以 introduce 另一 new PCI bus,可以容納更多的 PCI device.

► 05/06 - 05/13 (5)
► 04/29 - 05/06 (1)
► 04/22 - 04/29 (2)
► 2006 (1)

逛逛不一樣的地方

演算法 (影像處理, 資料結構, 智慧型視訊分析, 人工智慧)
平凡的幸福
相關資訊
流浪小築
旅遊美食~
小君君的祕密花園
繼續閱讀懶人加強版
幸福雅痞~
懷舊系列~

標籤

一些筆記 (10)
分享 (2)
心情分享 (3)
生活運用 (1)

P2P bridge亦有其 PCI config space,但是 lauout 與 PCI device有點不同,大家可以參閱P2P spec並與PCI device's config space比較一下.

在 P2P config space中,我常遇到的 issue是和下列 register有關的:
- Primary bus register: offset 18h
- Secondary bus register: offset 19h
- Subordinate bus register: offset 1Ah

Notes:
這三個 registers是BIOS在 PCI_SCAN時會決定的;所代表的意義是:這個 P2P bridge的上面 PCI bus number is ? 下面的PCI bus number is ? 及包含此 P2P bridge的 "branch" 最深的 PCI bus number is ?

Ex. 18/19/1Ah of one P2P bridge is 0/2/3
=> 此 P2P bridge 是 "bridge" PCI bus 0 and 2的(橋接在 PCI bus 0 and 2之間);而包含此P2P bridge的 PCI branch(想像成 tree structure) 最大(深)的PCI bus number is 3

- memory base/limit
- IO base/limit

Notes:
這兩個是 BIOS 在 PCI_SCAN時所 assign的. 所代表的是: resource "window" for devices behind this bridge.意即:若P2P bridge下面(就上例言:是 Bus 2上)有 PCI devices,則他們的 BARs 必須被包含在此 window 之內 !!!

[Practice]
[Q]假設有一個 P2P bridge ,下面有一PCI device;現在必須要去 accessPCI device's Device ID/Vendor ID(that is, PCI config read). 但是,問題是:做這事的"點"要在 PCI_SCAN之"前"....那要如何做到呢 ^_^ ?

Ans:假如要在很早前(Ex. PCI SCAN之前)去 access P2P bridge後面的 device,照理是做不到的,因為: P2P bridge沒有被正確的 configed...

在此例中,P2P bridge的 primary/secondary/subordinate bus要被 set,後面的device才能被 accessed到 !

所以,假如要在 PCI SCAN前就 access,則BIOS必須手動去 set 此 3 bytes;然後,PCI config access才能被 forward

to 其後的 PCI devices...

[Q] 如何 disable memory or IO resource window ?
Ans: 只要將 base設成比 limit "大" 即可 !!!

- 相關討論 Part1 -

前輩已經提到P2P Bridge我就直接問我的問題了

[Q1] Bridge 是用來擴充PCI Bus，在PCI Bus Spec與PCI-PCI Bridge Spec中定義，PCI Device是透過IDSEL來決定他的身分，其中PCI Bus Spec定義AD[31:11]，而PCI Bridge Spec定義AD[31:16] 當做IDSEL，這中間差異為何? 為什麼一個要從Bit 11開始當作是Dev 0，而另一個由Bit 16 才當作是 Dev 0 ?

Ans: 1. IDSEL對PCI device而言是 input,是用來當作 device's "chip-select"訊號. 而且,IDSEL "如何連接" 是 H/W 決定的,BIOS無法決定.

假如將板子上某個device's IDSEL "割斷",則此 device將無法接受 PCI configuration read/write(以 ru.exe來說,就是按F6後,是看不到它的...)

那要如何決定 device's IDSEL ? 一般而言, board designer會將 "unused AD lines"拿來做 IDSEL,以連接至 PCI device.

在 configuration access時,所下的 Ex. "o cf8 80001020"(看起來是 I/O transaction)會被 host bridge轉成 configuration transaction;此時, host bridge即可判斷此 transaction 要 access的 device是否在該 PCI bus上;if YES 轉成 Type 0 transaction;If NO 轉成 Type 1 transaction,並往下發送...(host bridge只要 check latched "bus information"即可完成此判斷 !)

以 Type 0言,AD bus上的 format as follows:
bit[1:0] = 00 (indicates "Type 0")
bit[7:2] indicates register number
bit[10:8] indicates function number

那 Bus number ? Device number ?
=> Bus number不必知道 ! 因為:Type 0產生即代表 bus number = 現在的 bus #
=> Device number呢 ? 因為,此時(Config transaction && address phase) AD bus bit[31:11]沒人用 !!! 因此, board designer會把此 21 bits拿來做 IDSEL用 !

因此, AD bus bit11 <-> device 0
12 <-> device 1
.....

當然,不可能 21 bits都拿來接 PCI devices;因為電路上的現實考量...

.....以上為:我所知為何從 bit11開始來當作 IDSEL.....

以 Type 1言,PCI-PCI bridge收到後,會將其 bus information與自己的 secondary bus number比較;若是 addressed device是在 secondary bus上,則將 Type 1 -> Type 0;若否,繼續包成Type 1往下一層送...

在P2P spec v1.1 page 22 有一張表,說明 IDSEL generation(from primary address -> secondary address),其中有提到: if primary address bit[15:11] =0,則 secondary address AD [31:16] = 0000 0000 0000 0001;以此類推.

所以,我覺得為什麼 for P2P bridge 其 IDSEL可由 bit[31:16] 來決定的原因在此 !!!(表的關係...)

.....以上為:我推論為何從 bit16開始來當作 IDSEL.....
[補充]
PCI config index register裡面的資料其實和硬體解出configuration cycle是相關的.
一.轉換出來是type 0 cycle的話. 硬體只要做以下兩件事.
1. mask 掉bus number(bit 16 ~ 31)以上的部份.
2.解碼 device number的部份即可到對應的 AD bit. 所以其最低可以使用的就是AD11.也就是說一個bus上最多只能有 21個 devices(只是由於推動力問題, 往往是做不到的).
Note:其實也可以設計成其他大於AD 11開始, 這要看chip設計者決定了.
二. 轉換出來是type 1 cycle的話. 只要做
1. mask 掉reserved以上的部份(bit 24 ~31)
2. bit 0 = 1
由於P2P跟其他device不同的地方就是, 除了type 0 clcye以外, 還必須處理 type 1 cycle. 這也是分成兩部份
一. type 1 -> type 0. 當 bus number 等於 secondary bus number 時候出現.
1. 解碼 device number 到對應的 AD. spec中有提到轉換的表. dev 0 = AD16....etc
2. 把 bit 0 由1 變成 0
二. type 1-> type 1. 當 bus number 介於 secondary bus number和 subordinate bus number
1. 直接往下一層送即可.交給其他的P2P 處理.

[Q2] 在IA32下，CONFIG_ADDRESS 會被轉成Configuration Cycles，當Bus Number <>0 時，NB會轉成 Type 1 然後往 DMI送到SB，當P2P Bridge收到後，然後定址到Slot上面的PCI Device，這樣說法對嗎?

Ans: 總而言之, 是自己local bus上的,就會轉成 Type 0,然後打在AD bus上,等待認領;若否就轉成 Type 1,往下一個bridge送,繼續尋找...對的人...for each bridge,都是一樣...

[補充]
對 PCI spec是, 如果以Intel PCI express架構來說. 那個已經被封裝成 pci express的 package了.沒有所謂的 type 0, type 1 cycle了.

[Q3]PCI Device透過IDSEL來決定身分，那PCIE Device呢? 我查過資料，好像PCIE不需要IDSEL那他是如何決定Device Number ?

Ans: 我所遇的 PCIe device也是由 AD bit[31:11]中找線拉至 device's IDSEL決定的.不知其他家 chipset是如何 implement.
[補充]
PCI express 是internal routing. PCI express是個跟PCI 完全不同的架構. 只是為了軟體相容性的關係, 把 software架構做的跟PCI bus一樣. PCI express是point-to-point架構, 一個link 只會連接一個device. 跟PCI 這種可以多個device在同一bus上是不一樣的. 所以 device number對PCI express是完全不重要的.
Note. AMD的Hyper transport 也是基於一樣的心態來設計軟體架構的.

※ PCIe 的device是 internal routing. 以規格來看,下一層的 device number都是為0.

- 相關討論 Part 2 -

DMI指的是 Intel 南北橋中間的通道 ! 之前也是不知P2P bridge部份關於IDSEL的配置,查了表才知道原來有這樣的 mapping(primary address<->secondary address). 其實,可以說 "unused AD bus 會被拿來當 IDSEL用"就是了吧 ^_^
[補充]
是的, 只要軟體能夠知道routing 關係.怎麼接都可以. 只要bus controller控好實際的IDSEL即可. P2P之所以會有嚴格規定(兩項, 1. IDSEL&device number表, 2. secondary bus IRQ routing)是因為P2P 不一定是在板子上. 包含卡都可以有P2P bridge. 在板子上的P2P 可以靠BIOS來建立正確的 routing, 但是插卡不行. 所以必須把這些定義好. 這樣 PnP software(BIOS or OS)才能正確的完成IRQ 分配.讓卡正常工作. 所以如果觀察某些板廠. 就算是真的p2p 沒有存在在板子上, 很多PCI slot的IRQ routing都是依據p2p spec裡面的規定做(因為SB的PCI bus還是落在P2P之後).

在 PCI scan時,BIOS會掃描整個系統的PCI architecture(包含 device & bridge);其掃描方式由BIOS's PCI kernel 來決定 !
[補充]
其實了解PCI spec. 要寫PCI scan其實可以效率好又正確. 常見的新進工程師寫法大概就是 3個 loop來處理. bus:0~255, device:0~31, function:0~7. 掃個 256*32*8次, 反正都是程式做, 結果往往也看來正確.這種寫法其實是不對的. (其實,若是多了解硬體的架構,就可以寫出有效率的code了！這也是F/W工程師的價值...)

Ex. Assume 系統架構是這樣的: NB,P2Px3, PCIe bridge x2;其中:
A. 3 P2Ps的配置 is: P2P0下面接P2P1;P2P下面接P2P2
B. PCIe x 2 & P2P0都在 bus 0;其PFA為
NB(0,0,0)
P2P0(0,1,0)
PCIE0(0,4,0)
PCIE1(0,5,0)

=> 最後的 PCI achitecture is:

Bus 0----------
NB(0,0,0),P2P0(0,1,0),PCIE0(0,5,0),PCIE1(0,6,0)

*下面 Bus 1/2/3由 P2P0/1/2所 introduce:
Bus 1----------
P2P1(1,0,0)
Bus 2----------
P2P2(2,0,0)
Bus 3----------

*下面 Bus 4由 PCIE0所 introduce:
Bus 4----------

*下面 Bus 5由 PCIE1所 introduce:
Bus 5----------

所以,Bus number 是由BIOS's scanning "algorithm"所決定的;假如採用 depth-first,則會產生上述的結果！決定後的值會填到 bridge的 Primary/secondary/subordinate bus number registers !

[Q]順便問個問題好了. 其實function number不應該是永遠需要scan的, 為什麼?什麼時候才需要scan function number?
Ans: 我想,對於 function number的問題,應該是: PCI header region offset 0Eh bit7代表: milti-function or NOT ! 因此,可以先 check此bit,再決定要不要往下掃了...這樣又少做了許多虛工...^_^

Ex. PFA (0,3,0) 有回應(that is, Vendor ID/Device ID != 0xFFFFFFFF),則先check (0,3,0)'s PCI Reg0Eh bit7; if "1" then 此device為 multi-function device,還要再往下找 Ex. (0,3,1~7) 有無回應;if "0" then try next device number...!

[補充，加快速的的方式]

檢查multi function bit是正確的, 但是不只是因為效率問題. 而是PCI 規格中, single function裝置可以不解碼 config cycle type 0 bit[8~10], 也就是說 一個 single function裝置, 會對 所有的function number回應, 也就是會出現 8 個相同的device.

順便說一下我的scan加速法. 其實我不是使用 vandor ID & device ID來判斷裝置存在與否. 我是用 class/subclass/interfae ID來作判斷. default 只scan bus 0, 遇到 P2P bridge才會把taget bus number+1, 如果遇到multi host(host bridge 數量> 1)的板子才會完整掃描 255個 bus. ^_^

~轉貼自艾克索夫實驗室~
Rootkit in PCI Option ROM

「Rootkit」一字來自 UNIX 界；但目前通常用於描述 Windows 木馬程式作者所運用的隱形技術。
起初，Rootkit指的是一組程式，可讓駭客躲過偵測。 為達成此目的，可執行的系統檔案 (如 login、ps、ls、netstat 等) 或系統程式庫 (libproc.a) 會遭到更換，或安裝核心模組。 這兩種動作只有一個相同目的；防止使用者收到正確資訊，知道電腦上發生了什麼事。

首先介紹PCI的基本常識, PCI Bus是在約1990年由Intel發展出來, 用來連接主機板上的各項裝置的匯流排標準, 後來成為業界的標準之一, Spec可以在PCI-SIG註冊會員之後下載, 架構上簡單的說, 就是一個Host bridge, 在一般的PC上通常指的就是North Brdige, 這個brdige後面就是bus#0(當然也有Multi Host-Bridge的狀況, 這邊舉例的是最單純的情況), 然後接到South Bridge, South Bridge之後可能接的是ISA Bus, IDE Controller, USB, IDE, DMA Controller等等, 如果bus#0上還有別的PCI Bridge, 這個Bridge後面就是bus#1, 如果有多個Bridge存在(PCI最多可以有256個bus), bus#就不一定是固定的了, 一個PCI Bus上可以有32個device, 每個device可以有8個function, 每個function都有屬於自己的256個register. 在PCI的規範裡, 256個register中的前0×40個是公定的功能, 從0×40到0xff則由各家廠商自行實作, 存取這些register的方法, 一般的PC上是透過IO port 0xCF8~0xCFF, 如果是新的PCI-Express則是直接透過Memory Mapped IO, 以存取記憶體的方式直接進行存取, 例如我們想要讀取一個在bus#0 dev#1 func#0的register#40~43, 透過IO的方式如下:

```
mov eax, (0x01 << 31 ) // Type-1 PCI Configuration
+ (0x00 << 16 ) // Bus#0
+ (0x01 << 11 ) // Device#1
+ (0x00 << 8 ) // Function#0
+ 0x40 // Register#40
mov dx, 0xCF8 // Index Port 0xCF8 ~ 0xCFB
out dx, eax
```

```
mov dx, 0xCFC // Data Port 0xCFC ~ 0xCFF
in eax, dx // Get Data in EAX
```

如果是透過MMIO的方式則簡單多了:

```
mov esi, (MMIO_BASE) + (0x00 << 20) + (0x01 << 15) + (0x00 << 12) + 0x40
mov eax, [esi]
```

今天我們比較有興趣的是位在0×30~0×33的register, 在PCI Spec定義中, 這裡的值存放的就是expansion rom在physical memory中被decode到IO的位址, 比如說這個位址是0xFE000000, 如果你在0×30把bit0設為0×1(io->mem decode), Command Register(0×04~0×05)的Memory Space Bit打開, 在0xFE000000的地方你就可以找到這個rom, 開頭是0×55AA(這當然也是規範之一, 用來辨視是否為一個PCI rom), BIOS在POST過程中, 會逐一掃描位於主機板上所有的PCI Device, 假如device上有rom, 就會把它給拷貝到memory中, 然後用jmp指令跳到ROM開頭offset 0×02(別忘了開頭offset 0×00是0×55AA)的地方開始執行PCI ROM, 執行完後ROM會再把控制權交回到BIOS手上, 一般而言, 在傳統記體空間中, 0xC0000~0xCFFFF是給VGA ROM用的, 0xD0000~0xEFFFF則是留給一般的PCI ROM使用, 當然各種情況下還是會有些許差異, 例如有些BIOS會保留0xE0000~0xFFFFF給自己使用, 像是BIOS的interrupt service, DMI data….等等雜七雜八的東西, 執行完的ROM仍然會保留在記憶體中, 因為有些ROM會修改IVT(Interrupt Vector Table), 將某些interrupt service導向自己的code, 像是VGA ROM可能就會hook int 0×10, 這是很合理的, 因為int 0×10是BIOS所提供用來控制螢幕的service, 聰明的你看到這裡應該就知道前面所提的那篇文章想說什麼了, 如果有個”惡意”的程式被埋在PCI ROM裡面, 只要一開機就會自動被執行, 它的運作並不是一時的, 它可以hook某個OS一定會用到的BIOS Interrupt Service, 然後在這個interrupt service被呼叫的時候動作就可以了, 而且麻煩的是即使你format你的HDD也沒用, 除非你把有問題的PCI Device從你的主機板上移除, 嗯….聽起來蠻炫的, 但是, 有可能嗎?

要回答這個問題之前, 需要知道一些基本的常識, 在保護模式下, 因為IO動作受到限制的關係, 要存取IO並不像在DOS那樣容易, 但如果想嘗試Re-flash一顆PCI ROM, 勢必得進行IO動作, 所幸在Windows下這並不是不可能的事, 有些人可能知道利用SeTcbPrivilege和使用ProcessUserModeIOPL structure呼叫undocumented Native API NtSetInformationProcess()就可以達成目的. 一旦攻擊者有辦法修改PCI ROM, 他就可以利用文章中所提到的例子: int 0×10(Windows在開機過程中會透過Ke386CallBios()呼叫int x010), 作他想作的任何事了.

可惜不論哪種攻擊方式, 最終都是要對OS kernel動手腳, 利用各種偵測工具(如: Archon Scanner), 一定可以找到有問題的地方, 如果最後的箭頭指向PCI ROM, 我們可以透過上文所述, 將存在PCI Device和memory中的ROM給dump出來, 需要dump兩邊的rom, 是因為PCI Spec規範中允許實際上所需要配置的記憶體不一定要等於原本rom的大小, 藉以節省保貴的記憶體(別忘了PCI ROM只能被配置到幾個64KB的segment裡而已), 然後向PCI Device的製造商索取正常版本的rom進行比對, 藉以得知是否為被修改的版本. 假如發現有不一樣的地方, 接下來可以朝幾個方向繼續分析是否為有問題的ROM, 我們可以檢查一下它是否修改了不必要的IVT, 像PXE ROM就不太可能hook int 0×10, 或是有保護模式相關的程式碼, 因為一般的ROM應該都是在real mode下執行, 所以應該不會切到protected mode, 如果有相關的程式碼那就非常可疑, 還有rom裡面是否有可疑的字串, 或是位在Windows Kernel位址空間裡的32-bit address, 另外ROM裡面也不太可能出現編碼過的code, 如果rom裡的code很難被

disassemble, 或是充滿了一堆obfuscated code, 那也是很有問題.

除了軟體的方式, 最近興起的新技術TPM也能克制這種攻擊手法

張貼者： 小華的部落格
標籤： BIOS相關

16 則留言:

poyuan 提到...

作者已經移除這則留言。

8月 21, 2008 1:13 上午

poyuan 提到...

BIOS界的前輩您好, 無意間看到您的部落格, 對於您巨細彌遺的文章, 深為感動. 謝謝您在工作閒聊之餘, 寫了這些BIOS相關的文章.
有一個問題想請教您: 敝公司做了一塊PCI卡,發生了難以理解的現象, 就是在P915的主機板上可以WORK, 但是在P35 or P965的主機板上卻不行?!
用LA側錄PCI BUS的訊號後發現, BAR0被WRITE 0x80008000, 再被READ出 0x80008001, 最後BAR0被WRITE
0xFFFF_FFFF, 等於BAR0是被關閉? 故無法map 到IO SPACE? 查了很久不知道原因, 可不可以請前輩給一點HINT?

8月 21, 2008 1:15 上午

小華的部落格 提到...

我進入BIOS才兩年,很多東西不懂! 大家互相討論啦!

關於你的問題,我不太懂你測錄的訊號,

不過我從PCI Spec 看到的步驟是:

1.對BAR寫入"全部1"的值
2.讀回BAR狀態值並判斷,假設bit 0=1 則代表這個PCI應該是實做IO Space

3.從bit 0開始往高位元檢查第一個"1"的出現的權值..假設出現在bit 8 , 代表這是一個256 bytes的IO範圍

4.填入IO BaseAddress (因為是256,所以填入的bit7:0無效)

所以解碼範圍是IOBASE~IOBASE+256

5.到CMD Register去Enable BAR

所以你看到寫入FFFF_FFFF應該是我說的步驟1...所以你可能還是要繼續測錄一下後面的訊號,希望這些資訊能幫的上你的忙!

8月 25, 2008 8:58 上午

Colorben 提到...

個人問題請教：我要讀取的rtl8139網卡都很順利讀到mac addr，在p35主板上內建好像是rtl8168也都從BAR就可以順利讀取，因為它们都是I/O mapping，但從ga-P965-ds3主版上內建網卡讀取mac addr確讀不到正確ㄉ值，因為它是memory mapping，從bar讀到ㄉ值是f4000004,像這樣我要如何正確讀取網卡mac addr?

3月 17, 2009 8:19 上午

Colorben 提到...

我目前是在DOS下寫ㄉ一個應用，由於必須得知網卡MAC ADDRESS，所以我寫ㄌ這樣ㄉROUTINE：

```
subGetMacAddr proc uses eax ecx dx si di
xor ebx,ebx
xor si, si ;index 0
mov ecx,020000h ;因為網卡ㄉclass=02h Subclass=00h
mov ax, 0B103h ;find PCI B/D/F
int 1ah
jc loc_err
mov di, 10h ;register 10h
mov ax, 0B10Ah ;read pcicfg dword
int 1ah
test ecx,1
jz Memoryio
mov dx, cx ;port
xor dl, dl
in eax, dx
out 0EDh, al
```

mov DWORD PTR MacAddr, eax
add dx, 4
in ax, dx
out 0EDh, al
mov WORD PTR MacAddr[4], ax
loc_err:
ret
Memoryio:
我想在這裡加入若網卡的BAR是memory mapping，就使用MMIOㄉ方式讀取此時ㄉECX：
0x0f4000004(marvell yukon 88E8056 配置BUS4 DEVICE0 FUNCTION0)
ret
subGetMacAddr endp

由於我只會碰到兩塊主機板：GA-965P-DS3及GA-P35-DS3L，GA-965P-DS3ㄉ主機板所配置ㄉ網卡是marvell yukon 88E8056 ㄉController(它是memory mapping)，另外GA-P35-DS3L配置ㄉ網卡是RTL8168 Controller(它是IO映射)，上面那ㄍROUTINE對RTL8168可以讀取到MAC ADDRESS，而且外接RTL8139也可以讀到MAC，會用上面ㄉ方法也是因為在網路上找到如附圖ㄉ資料。但是marvell yukon 88E8056沒辦法讀到MAC，紅色區域就是想為它而寫，不知是否能幫幫我。
我使用過以下方法但都讀不到MAC ADDR
mov esi, (MMIO_BASE) + (0x04 << 20) + (0x00 << 15) + (0x00 << 12) + 0x10
mov eax, [esi]
當然我知道這和實作硬體有關，況且又沒有SPEC，所以只能求救ㄌ。
TKS,Benson

4月 22, 2009 11:19 下午

Birdy 提到...

請問前輩, 最近公司要開發一個板子, 是有6個PCI slot. 3個slot後會接一個PCI-to-PCI bridge, 然後再擴3個slot出來.
但另外有一片PCI週邊卡片, 上面是由一個PCI-to-PCI bridge去接4個PCI LAN.
請問當把這片週邊接到第4-6的slot時, 上面的PCI LAN能動嗎? 若第3個slot接了這片有PCI-to-PCI bridge的卡後, 第4-6的slot又接一片這個LAN卡, 這樣子兩片都能動嗎?拜託幫我解答一下吧? 如果要能動, PCI SCAN與BUS的關係是怎樣呢? 謝謝!

4月 28, 2009 10:54 上午

匿名 提到...

我現在在做的Project為AMD CPU+NB RS780+SB SB710.

掛在NB下有三個PCIE devices
兩個Lan chip 和一個Mini PCIE slot

當我Clear CMOS 後的第一次開機
系統可是Scan 到兩個LAN chip
但Scan不到Mini PCIE slot(Wireless lan)

第二次(CMOS 儲存)開機時
則三個devices都抓不到(RU看不到)

有人可以幫忙嘛?
我目前用AMI core8

Grant
PS. email : Grantwu00@yahoo.com.tw

2月 19, 2010 4:02 下午

Grant 提到...

Hi Harrison,

因為我現在在一個17人的小公司
公司只有我一個BIOS engineer
我一人孤軍奮戰實在很辛苦

如果你方便給我你的Email
我好和您討論問題

謝謝
Grant

2月 19, 2010 11:46 下午

匿名 提到...

我是搞OS Kernel的, 小華大的文章真的對我幫助很大, 因為很多基礎的東西不太可能一直纏著 BIOS team 去問.

最近 trace 過 log and code, 對照 PCI System Architecture 書, 再看到小華大的文章就更有感覺了, 很多事情得到 BIOS 這邊的映証.

感恩

3月 01, 2012 11:42 下午

klhsieh 提到...

作者已經移除這則留言。

3月 22, 2012 1:41 上午

胖皮(PongPeey) 提到...

作者已經移除這則留言。

8月 25, 2013 2:15 上午

phssub 提到...

請問有人會用devC++ 寫PCI dump出bus0,device0,function0的程式嗎?

8月 25, 2013 2:19 上午

eric 提到...

你好，請問你知道如何讀取XEON平台下DRAM的SPD資料嗎?
謝謝。

12月 22, 2013 9:00 下午

e398i 提到...

您好，不好意思想問一個比較新手的問題，
請問要如何在Shell 環境底下用C語言寫出一個PCI SCAN 的程式呢?
我的想法是寫迴圈然後讓他跑到我要讀得值停下來，
之後顯示出來，但是用迴圈存取PCI的概念我不是很了解...，
能否給個提示呢?謝謝!

8月 30, 2015 6:50 下午

Louis 提到...

To e398i

我記得網路上就有不少的範例嘍

而且有些Function可以直就選擇bus dev func

9月 02, 2015 3:33 上午

ZHY 提到...

你好，请教两个问题，
第一个问题是PCI segment(也有叫PCI domain的，就是MCFG里面配置的选项)的作用究竟是支持多个Host Bridge，还是在一个Host Bridge里面支持多个Root Bridge呢?
第二个问题是如果有了多个PCI segment，那么这些segment对应的Bridge是共享一个南桥(比如ICH)呢，还是每个segment里面都有自己的南桥?
初学者不甚了解，提前感谢!

2月 28, 2018 2:38 上午

張貼留言

較新的文章 首頁 較舊的文章