BIOS Practice: PCI Device Enumeration 1

原创 Anthony Modified on 2022-03-01 14:05:36 Read 2.8k Collection 13 Likes 1

Category Column: BIOS learning practice Article Tags: UEFI C

PCI device enumeration is mainly performed in two different ways, one is IO, and the other is the pciio protocol in UEFI. Let's first look at the first one.

PCI device enumeration using IO

First, let's write a function to read the PCI configuration space. This is written in assembly and is easy to understand:

AI generated projects 登录复制

```
unsigned long PciRead(unsigned long index)
{
    unsigned long pcidata;
    //asm .386              //use 32bit registers
    _asm xor eax, eax      //clear eax reg
    _asm xor ebx, ebx      //clear ebx reg
    _asm mov eax, index    //mov index address to eax
    _asm mov dx,  0x0CF8   //mov PCI index reg to dx
    _asm out dx,  eax      //write index address to index reg
    _asm mov dx,  0x0CFC   //mov PCI data reg to dx
    _asm in  eax, dx       //read data from data reg
    _asm mov pcidata,eax   //mov data to pcidata
    return  pcidata;
}
```

收起

With this function, let's move on to the main text:

AI generated projects 登录复制

```
void main()
 {

   unsigned long i,bus,dev,func,pci_addr,pci_data,VendorID,DeviceID,SSID,SVID,ClassCode;
   printf("BusNo DevNo FuncNo DevID VenID SSID SVID Describe\n");

   for(bus=0;bus<256;bus++)
      {
         for(dev=0;dev<32;dev++)
           {
              for(func=0;func<8;func++)
                {
                   pci_addr=0x80000000+(bus<<16)+(dev<<11)+(func<<8);
                   pci_data=PciRead(pci_addr);

                   VendorID=pci_data & 0xffff;
                   //判断PCI设备是否存在
                    if(VendorID!=0xffff && VendorID!=0){

                    DeviceID=(pci_data>>16)& 0xffff;

                    index=(index&0xffffff00)+0x2c;
                    pci_data=PciRead(index);
                    SSID=(pci_data>>16)& 0xffff;
                    SVID=pci_data & 0xffff;

                    index=(index&0xffffff00)+0x08;
                    pci_data=PciRead(index);
                    ClassCode=(pci_data>>24) & 0xffff;
                   }
              }
         }
   }
}
```

收起

Of course, the program logic is written, but the whole is not yet complete. What is this Classcode? It is the function category number. Let's take a look at the UEFI code:

AI generated projects 登录复制

```
//
  // Check whether this is IDE
  //
  if ((PciData.Hdr.ClassCode[2] != PCI_CLASS_MASS_STORAGE) ||
      (PciData.Hdr.ClassCode[1] != PCI_CLASS_MASS_STORAGE_IDE)) {
    return ;
  }
```

Ok, then we create an array ourselves to correspond to the categories:

AI generated projects 登录复制

```
char *Describe[20]={   "Function built before class codes were defined",
                        "Mass storage controller.",
                        "Network controller.",
                        "Display controller.",
                        "Multimedia device.",
                        "Memory controller.",
                        "Bridge device.",
                        "Simple communications controllers.",
                        "Base system peripherals.",
                        "Input devices.",
                        "Docking stations.",
                        "Processors.",
                        "Serial bus controllers.",
                        "Wireless controllers.",
                        "Intelligent IO controllers.",
                        "Satellite communications controllers.",
                        "Encryption/Decryption controllers.",
                        "Data acquisition and signal processing controllers.",
                        "Reserved.",
                        "Device does not exist.",
                    };
```

收起 ^

Now that we have everything, let's start working on the complete code.

```
#include<stdio.h>
#include<conio.h>

unsigned long PciRead(unsigned long index);

unsigned long PciRead(unsigned long index)
{
    unsigned long pcidata;
    //asm .386              //use 32bit registers
    _asm xor eax, eax      //clear eax reg
    _asm xor ebx, ebx      //clear ebx reg
    _asm mov eax, index    //mov index address to eax
    _asm mov dx,  0x0CF8   //mov PCI index reg to dx
    _asm out dx,  eax      //write index address to index reg
    _asm mov dx,  0x0CFC   //mov PCI data reg to dx
    _asm in  eax, dx       //read data from data reg
    _asm mov pcidata,eax   //mov data to pcidata
    return  pcidata;
}

char *Describe[20]={   "Function built before class codes were defined",
                        "Mass storage controller.",
                        "Network controller.",
                        "Display controller.",
                        "Multimedia device.",
                        "Memory controller.",
                        "Bridge device.",
                        "Simple communications controllers.",
                        "Base system peripherals.",
                        "Input devices.",
                        "Docking stations.",
                        "Processors.",
                        "Serial bus controllers.",
                        "Wireless controllers.",
```

```
                      "Intelligent IO controllers.",
                    "Satellite communications controllers.",
                    "Encryption/Decryption controllers.",
                    "Data acquisition and signal processing controllers.",                      "Reserved.",
                      "Device does not exist.",
                   };

void main()
 {
   char *p;
   unsigned long i,bus,dev,func,pci_addr,pci_data,VendorID,DeviceID,SSID,SVID,ClassCode;
   printf("BusNo DevNo FuncNo DevID VenID SSID SVID Describe\n");

   for(bus=0;bus<256;bus++)
      {
         for(dev=0;dev<32;dev++)
           {
              for(func=0;func<8;func++)
                {
                   pci_addr=0x80000000+(bus<<16)+(dev<<11)+(func<<8);
                   pci_data=PciRead(pci_addr);

                   VendorID=pci_data & 0xffff;
                   //判断PCI设备是否存在
                    if(VendorID!=0xffff && VendorID!=0){

                    DeviceID=(pci_data>>16)& 0xffff;

                    index=(index&0xffffff00)+0x2c;
                    pci_data=PciRead(index);
                    SSID=(pci_data>>16)& 0xffff;
                    SVID=pci_data & 0xffff;

                    index=(index&0xffffff00)+0x08;
                    pci_data=PciRead(index);
                    ClassCode=(pci_data>>24) & 0xffff;

                    p=Describe[ClassCode];
                    printf(" %02x %02x %02x %04x %04x %04x %04x %s\n",bus,dev,func,DeviceID,VendorID,SSID,SVID,p);

                   }
             }
         }
   }

}
```


Note: The code is just a general summary of the logic. The specific implementation depends on each platform (I don't know if the assembly can be read...)