BIOS Practice: PCI Device Enumeration 2

原创 Anthony Modified on 2022-03-01 14:05:18 Read 2.8k Collection 10 Likes 1

Category Column: BIOS learning practice Article Tags: C UEFI

摘要 This article introduces two ways to enumerate PCI devices in UEFI BIOS: through IO enumeration and through pciio enumeration. It also describes in detail how the PCIHostBridge controller driver specifies I/O space and Memory space for PCI devices, and discovers PCI devices through the PCI bus driver.

The summary is generated in C Know , supported by DeepSeek-R1 full version, go to experience>

Last time I mentioned that in addition to IO enumeration of PCI devices, there is another way to enumerate PCI devices, that is through pciio, which is also a common operation in UEFI code (of course there is also MMIO, but I won't write about it)

UEFI BIOS provides two main modules to support PCI bus, one is PCI Host Bridge controller driver, the other is PCI bus driver.

The PCI Host Bridge controller driver is bound to a specific platform hardware. According to the actual IO space and memory map of the system, it specifies the range of I/O space and Memory space for PCI devices, and generates PCI Host Bridge Resource Allocation protocol for use by PCI bus drivers. The driver also generates handles for all RootBridge devices under the HostBridge controller, and PciRootBridgeProtocol is installed on the handles. The PCI bus driver uses PciRootBridgeIo Protocol to enumerate all PCI devices in the system, discover and obtain the Option Rom of the PCI device, and calls PCI Host Bridge Resource Allocation protocol to allocate PCI device resources. The PCI RootBridge device generates PCI Local Bus. PCI device drivers do not use PCI Root Bridge I/O protocol to access PCI devices, but use PCI IO Protocol generated by the PCI bus driver for PCI devices to access PCI IO/MEMORY space and configuration space.

After reading the above, let's get straight to the point:

```
EFI_STATUS
EFIAPI
ShellAppMain (
  IN UINTN Argc,
  IN CHAR16 **Argv
  )
{
  EFI_STATUS                  Status = EFI_SUCCESS;
  EFI_HANDLE                  *HandleBuffer;
  UINTN                       PciController_Count, Seg, BufferSize=0;
  UINTN                       NumHandles, i;
  EFI_PCI_IO_PROTOCOL         *PciIoProtocol;
  UINT8                       ListDevice = 0, SaveOpRom=0;
  UINT8                       IndexOfSavedDevice=0;
  PCI_CONTROLLER_INFO         PciController_Info[50];
  UINT32                      VenderDevId;
  CHAR16                      SaveFileName[100];

  Print(L"================================================\n");
  PciController_Count = 0;

  if(Argc >= 2)
  {
    if(StrCmp(Argv[1], L"-L")==0)
    {
      ListDevice = 1;
    }else if(StrCmp(Argv[1], L"-S")==0)
    {
      if(Argc!=3)
      {
        Status = EFI_INVALID_PARAMETER;
        Print(L"Please Specify Index of Device when save OpRom\n");
        goto ProcExit;
      }else
      {
        IndexOfSavedDevice = StrDecimalToUint64(Argv[2]);
        SaveOpRom = 1;
      }
    }
  }else
  {
    Print(L"Too few parameters\n");
    Status = EFI_INVALID_PARAMETER;
```

```
    goto ProcExit;
                        }

Status = gBS->LocateHandleBuffer (ByProtocol, &gEfiPciIoProtocolGuid, NULL, &NumHandles, &HandleBuffer);

  for(i=0; i< NumHandles; i++)
  {
    Status = gBS->HandleProtocol(HandleBuffer[i], &gEfiPciIoProtocolGuid, (void**)&PciIoProtocol);
    if(!EFI_ERROR(Status)){
      if(PciIoProtocol->RomSize>0){
        VenderDevId = 0xFFFFFFFF;
        PciIoProtocol->Pci.Read(PciIoProtocol, EfiPciIoWidthFillUint32, 0, 1, &VenderDevId);
        Seg = 0;
        PciIoProtocol->GetLocation(PciIoProtocol,
                                   &Seg,
                                   &PciController_Info[PciController_Count].Bus,
                                   &PciController_Info[PciController_Count].Device,
                                   &PciController_Info[PciController_Count].Func);

        PciController_Info[PciController_Count].Handle =   HandleBuffer[i];
        PciController_Info[PciController_Count].PciIo = PciIoProtocol;
        PciController_Info[PciController_Count].VidDid = VenderDevId;
        PciController_Count +=1;
      }
    }
  }


  if(ListDevice)
  {
    Print(L"Controller With OpRom Number: %d \n",PciController_Count);
    for(i=0;i<PciController_Count;i++)
    {

      Print(L"Controller ID: %d  VidDid: %08x  Bus: %x  Dev: %x Func: %x  Size: 0x%x\n",
                        i,
                        PciController_Info[i].VidDid ,
                        PciController_Info[i].Bus,
                        PciController_Info[i].Device,
                        PciController_Info[i].Func,
                        PciController_Info[i].PciIo->RomSize);
    }
....
```


I have only posted part of the code, but it is enough. You can write an app yourself with the above code. The code is very easy to understand.