UEFI Development Exploration 71- YIE001PCIe Development Board (07 OptionROM Framework)

原创 luobing4365 Posted on 2021-01-17 17:52:10 Read 1.3k Collection 4 Likes 3 copyright
Category Column: UEFI Development Article Tags: uefi BIOS Option ROM Low-level application development EfiRom

UEFI Development This column includes this content 503 Subscribe 104 articles Subscribe to our column

(Please keep it-> Author: Luo Bing https://blog.csdn.net/luobing4365)

I am currently editing a new book "UEFI Programming Practice" (tentative title). The content introduced now has been organized into Chapter 8.

Writing a book is quite time-consuming. I can write freely in a blog, but I cannot do that with a book manuscript. In order to be accurate, sometimes I need to spend the whole morning looking for information to confirm a data. I hope it can be published in the first half of this year.

As for Option ROM, its history and principle have been described in Chapters 34, 35, and 36. Now that we have a development board, we can start development directly on it.

1Develop Option ROM

The code is developed based on the architecture in BlankDrv, and mainly modifies the Supported() function and the Start() function .

The modified Supported() code is shown in Example 1.

Example 1 : Supported() function

```
EFI_STATUS EFIAPI BlankDrvDriverBindingSupported (
  IN EFI_DRIVER_BINDING_PROTOCOL *This,
  IN EFI_HANDLE Controller,
  IN EFI_DEVICE_PATH_PROTOCOL *RemainingDevicePath OPTIONAL)
{   EFI_STATUS Status;   UINT16 MyVendorID, MyDeviceID; EFI_PCI_IO_PROTOCOL *PciIo;   Status = gBS->OpenProtocol (Controller,
&gEfiPciIoProtocolGuid,                  (VOID **) &PciIo,This->DriverBindingHandle,                  Controller,EFI_OPEN_PROTOCOL_BY_DRIVER);   if
(EFI_ERROR (Status)) return Status;   Status = PciIo->Pci.Read(PciIo,EfiPciIoWidthUint16,0,1,&MyVendorID);   if (EFI_ERROR (Status)) goto Done; //Get
vendor ID   Status = PciIo->Pci.Read(PciIo,EfiPciIoWidthUint16,2,1,&MyDeviceID);   if (EFI_ERROR (Status)) goto Done; //Get device ID   Status =
EFI_SUCCESS;   //Based on vendor ID and device ID, determine whether it is the target device   if (MyVendorID != CH366_VENDOR_ID || MyDeviceID !=
CH366_DEVICE_ID)     Status = EFI_UNSUPPORTED;   Done: //Close the used Protocol   gBS->CloseProtocol(Controller,&gEfiPciIoProtocolGuid,         This-
>DriverBindingHandle,Controller);   return Status; }
```

In the Supported() function, first use OpenProtocol() to open the PCI I/O Protocol. If the opening fails, EFI_UNSUPPORTED is returned. Then use the obtained PCI I/O Protocol instance to obtain the manufacturer ID and device ID from the configuration space of the PCI device.

The modified Start() function is shown in Example 2.

[Example 2] Start() function

```
EFI_STATUS EFIAPI BlankDrvDriverBindingStart (
  IN EFI_DRIVER_BINDING_PROTOCOL *This,
  IN EFI_HANDLE Controller,
  IN EFI_DEVICE_PATH_PROTOCOL *RemainingDevicePath OPTIONAL )
{   …… //Code omitted   Status = PciIo->Pci.Read(PciIo,EfiPciIoWidthUint16,16,1,&MyIoBaseAddr);       if (EFI_ERROR (Status)) goto Done;   MyIoBaseAddr=
(MyIoBaseAddr&0x0fffe); //Get I/O base address HelloUEFI(); //Oprom demonstration function   Status = EFI_SUCCESS; Done: …… //Code omitted return
Status; }
```

At the beginning of the function, the Start() function obtains the base address in the PCI space of the device through the instance of PCI I/O Protocol. Then the HelloUEFI() function is called to implement the required functions.

That is to say, after completing these frameworks, you can focus on writing the HelloUEFI() function. In the series of blogs on UEFI development and exploration, various protocols are used to implement sample programs of various functions. These programs can be ported to this place with slight modifications.

In this article, we will add a print string directly in HelloUEFI() and wait for user input to implement a simple demonstration function. As shown below:

```
VOID HelloUEFI(VOID)
{   gST->ConOut->OutputString(gST->ConOut,L"Hello, I am YIE001!\n\r");   WaitKey(); //Wait for the user to press a key }
```

2 Compile and test Option ROM

The UEFI driver you write needs to be compiled according to a certain method to generate Option ROM. In Chapter 36, we have introduced how to compile in detail.

This article uses the method of modifying the INF file to compile the UEFI driver. Example 3 gives a sample of the INF file modification.

[Example 3] INF file

```
[Defines]
  ... //Other variables, omitted
  PCI_VENDOR_ID = 0x1C00 //Vendor ID
  PCI_DEVICE_ID = 0x4349 //Device ID
  PCI_CLASS_CODE = 0x020000 //Device classification number
  PCI_REVISION = 0x0003 //Code version
  PCI_COMPRESS = TRUE //Whether to compress, TRUE means compression
```

Assuming that the sample project you wrote is MyOprom, the compilation command is as follows:

```
C:\UEFIWorkspace>build -t VS2015x86 -p RobinPkg\RobinPkg.dsc \
-m RobinPkg\Drivers\MyOprom\ MyOprom.inf -a X64
```

Note that you should compile using the x64 target architecture, as most machines have 64-bit UEFI BIOS.

After the compilation is completed, two files, MyOprom.efi and MyOprom.rom, will be generated in the output directory. Use the EfiRom tool and run the "-d" command to view the information of MyOprom.rom. You can see that MyOprom.rom is already a valid UEFI Option ROM file, and its manufacturer ID, device ID and other information are consistent with those set in the INF file.

There are two ways to test. One is to use the load command or loadpcirom command in UEFI Shell to directly mount the driver on the handle of the device controller to display the functions implemented in MyOprom. The other is to write the binary ROM file into the Flash ROM on YIE001 in binary form according to the requirements of CH366 for testing.

Both of these test methods require the YIE001 development board to be plugged into the actual machine and run.

Here we will first introduce the former test method, and the latter method will be discussed in the next blog of YIE001.

Copy the compiled 64-bit efi program and rom files, MyOprom.efi and MyOprom.rom, to the UEFI boot USB drive.

Insert the development board YIE001 into the PCIe slot of the motherboard, then start the test computer and enter the UEFI Shell environment.

In the UEFI Shell environment, you can use the Shell command "pci" to list all PCI devices and check whether the development board YIE001 is recognized by the system. The manufacturer ID set on the YIE001 development board is 0x1C00, and the device ID is 0x4349. You can find out whether it exists in the listed PCI devices.

Use the following command to test:

```
load MyOprom.efi
```

or

```
loadpcirom MyOprom.rom
```

After the driver (or Option ROM) is loaded, the screen will display the string printed by the HelloUEFI() function in the sample project.