

UEFI Principles and Programming Practice--Analysis of Graphics Card Driver (UEFI Direction) of UEFI Driver Model

原创

Anthony


🕒 Posted on 2022-01-12 11:52:52

👁 Read 4.3k

🌟 Collection 12

👍 Likes 3

Category Column: [UEFI](#) Article Tags: [UEFI](#)

 **UEFI** This column includes this content

23 articles

[Subscribe to our column](#)

If you disassemble a domestic computer, you will find that many important chips have actually been domestically produced, such as CPU, GPU, **power management chip**, etc. I just took some notes on GOP. Looking back, I have never paid much attention to the operating logic of the graphics card drivers included in the manufacturer. So, this article will explore in depth how this graphics card driver is made.

Among the domestic graphics cards, Jingjiawei graphics cards are at the forefront, so this time we will analyze how the JM7200 driver is made.

UEFI driver model

There are two types of drivers, one that complies with the driver model and the other that does not. The core of the UEFI driver model is to manage the driver through the EFI Driver Binding Protocol. As a user-friendly driver, it usually also contains an EFI Component Name Protocol. In the entry function of the UEFI driver, the location where the EFI Driver Binding Protocol is installed is usually its own Handle. This instance will reside in memory for driver installation and uninstallation. Pasting the code directly violates the confidentiality agreement, so I will use the USB driver code to explain it.

UEFI driver loading process

UEFI will call gBS->StartImage(..) to execute the entry function of DriverImage. In the entry function, Driver Binding Protocol is loaded into Handle (Driver Imagehandle or other Controller Handle). Then UEFI will traverse all controllers and call CoreConnectSingleController function for each controller. In CoreConnectSingleController, EDBP's Supported function will be called to test whether the driver supports the controller. If supported, Start() will be called to install the driver.

The role and composition of EFI Component Name Protocol

Usually each driver also has a printable name to display driver information to the user. This printable name is provided by the EFI Component Name Protocol (ECNP) or EFI Component Name2 Protocol (ECN2P). ECNP and ECN2P are not required protocols for drivers, but it is recommended that driver developers provide this protocol.

AI generated projects 登录复制

```
1 EFI_STATUS
2 EFIAPI
3 UsbBusDriverEntryPoint (
4     IN EFI_HANDLE      ImageHandle,
5     IN EFI_SYSTEM_TABLE *SystemTable
6 )
7 {
8     EFI_STATUS      Status;
9
10    Status = EfiLibInstallDriverBindingComponentName2 (
11        ImageHandle,
12        SystemTable,
13        &mUsbBusDriverBinding,
14        ImageHandle,
15        &mUsbBusComponentName,
16        &mUsbBusComponentName2
17    );
18 }
19 ASSERT_EFI_ERROR (Status);
20
21 Status = gBS->InstallMultipleProtocolInterfaces (
22     &ImageHandle,
23     &gEfixxxxxGuid,
24     &xxxxxxx,
25     NULL
26 );
27 ASSERT_EFI_ERROR (Status);
28
29 return Status;
30 }
```

收起 ^

The above code is the basic operation flow of the entry function, which installs EDBP and ECNP, and then installs the required protocols, such as **the graphics driver** 's own protocol.

EDBP has three functions. Let's start with VideoControllerDriverSupported. For the graphics card, it is a PCI device, so PciIoProtocol is needed here. Let's open this protocol and get the VID and DID of the PCI device to see if they match. If they do, we start to implement the Start service.

AI generated projects

登录复制

```
1 //
2 // Open the PCI I/O Protocol
3 //
4 Status = gBS->OpenProtocol (
5     Controller,
6     &gEfiPciIoProtocolGuid,
7     (VOID **) &PciIo,
8     This->DriverBindingHandle,
9     Controller,
10    EFI_OPEN_PROTOCOL_BY_DRIVER
11    );
12
13 if (EFI_ERROR (Status))
14 {
15     return Status;
16 }
17 //
18 // Read the PCI Configuration Header from the PCI Device
19 //
20 Status = PciIo->Pci.Read ( PciIo, EfiPciIoWidthUint32, 0, sizeof (Pci) / sizeof (UINT32), &Pci );
21
22
23 Pci.Hdr.VendorId==?
24 Pci.Hdr.DeviceId==?
```

收起 ^

After entering the Start service, the graphics card-specific data is generally set and obtained first, and then the video module buffer is constructed, which is actually setting some video-related data.

AI generated projects

登录复制

```
1 VideoMode = &JM5400VideoVideoModes[0];
2 for (Index = 0; Index < JM5400_VIDEO_MODE_COUNT; Index++)
3 {
4     ModeData->ModeNumber = Index;
5     ModeData->HorizontalResolution = VideoMode->Width;
6     ModeData->VerticalResolution = VideoMode->Height;
7     ModeData->ColorDepth = VideoMode->ColorDepth;
8     ModeData->RefreshRate = VideoMode->RefreshRate;
9     DEBUG ((EFI_D_INFO, "Adding Video Mode %d: %dx%d, %d-bit, %d Hz\n",
10         ModeData->ModeNumber,
11         ModeData->HorizontalResolution,
12         ModeData->VerticalResolution,
13         ModeData->ColorDepth,
14         ModeData->RefreshRate
15     ));
16     ....
```

收起 ^

Now that the data is ready, it is time to construct the image:

AI generated projects

登录复制

```
1 GraphicsOutput = &Private->GraphicsOutput;
2 GraphicsOutput->QueryMode = JM5400VideoGraphicsOutputQueryMode;
3 GraphicsOutput->SetMode = JM5400VideoGraphicsOutputSetMode;
4 GraphicsOutput->Blt = JM5400VideoGraphicsOutputBlt;
5 .....
```

How the specific content is constructed is the core content. Of course, this section is just to familiarize yourself with how the graphics card driver is done.

