UEFI Development Exploration 76- YIE001PCIe Development Board (09 Interface and Keyboard Control)

原创 luobing4365 Posted on 2021-02-13 09:42:32 Read 650 Collection 1 Likes 1 copyright

Category Column: UEFI Development Article Tags: UEFI PCI-E Option ROM Low-level application development YIE001

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)

YIE001PCIe development board interface and keyboard control

Developing on the YIE001 development board is actually no different from developing UEFI applications in the previous blog. What needs to be paid attention to is whether the UEFI mechanism and Protocol used can be used normally when the Option ROM is loaded by the BIOS.

Generally speaking, GraphicsOutput Protocol and ConOut Protocol can be used normally. However, other protocols, such as creating Events, are difficult to say. On the machine I am currently testing, when loading Option ROM, the created Events do not work as expected. However, the same code works normally when tested under the UEFI Shell of the same machine.

The programming process of this code is as follows.

1 Graphics support and Chinese character support

The codes in UEFI development exploration, especially the graphics and Chinese characters, are transplanted from my open source project Foxdisk. They are modified according to the UEFI mechanism. Therefore, the codes in the blog with the same file name have similar contents.

Of course, as each project progresses, the code is constantly being modified. Even for me, it is difficult to tell when the file I am using was modified. For example, recently, a non-blocking function CheckKey() was added to the source file that handles the keyboard.

I try to maintain a loose but logically consistent structure, so that source files can be used directly. Code that processes the same hardware or the same logical layer will basically be in the same source file.

The functions of the source files are listed below: **Table 1 Functions of the source files**

Source Files	Pin Name
Graphic.c, Graphic.h	Image Processing
Keyboard.c, Keyboard.h	Keyboard handling
Window.c, Window.h	Interface related code, such as background settings, etc.
Font.c, Font.h	Text display, including Chinese characters and English
Common.c, Common.h	Provides various Protocol instances and some common functions

Copy the above files to the folder of the Option ROM framework code built previously, modify the corresponding INF file, and you can use it directly.

The display mechanism of Chinese characters has been discussed a lot in previous blogs. You can check out the previous blogs and codes.

However, the topics discussed in the blog are still quite scattered. In the book "UEFI Programming Practice", these topics are organized into chapters for compilation and reading. Considering copyright issues, the extraction tools of various Chinese character libraries are also rewritten. When the new book is released, I will share the code and tools on Gitee and GitHub, and those who are interested in technology can download and use them directly.

2 Interface and keyboard control programming

The Option ROM code can only run after it is loaded into the BIOS memory. In this regard, the development board YIE001 and YIE002 are very different. After all, YIE001 is not an independent MCU and cannot run independently.

The code in this article links the keyboard control with the hardware control of YIE001. The specific implementation code is as follows:

```
VOID HelloUEFI(VOID)
{
  UINT64 flag;
  EFI_INPUT_KEY key={0,0};
  UINT8 *s_text = "Alasse' aure,";  //《魔戒》精灵语的 “你好，日安”
  UINT8 *s_text1 = "欢迎进入UEFI的世界! ";
  UINT8 *s_text2 = "按'ESC'键退出此界面";
  flag = InintGloabalProtocols(GRAPHICS_OUTPUT);
  Print(L"flag=%x\n",flag);

  //图形显示测试
  SwitchGraphicsMode(TRUE);
  SetBKG(&(gColorTable[DEEPBLUE]));

  draw_string(s_text, 110, 60, &MyFontArray, &(gColorTable[WHITE]));
  draw_string(s_text1, 80, 100, &MyFontArray, &(gColorTable[WHITE]));
  draw_string(s_text2, 135, 140, &MyFontArray1, &(gColorTable[YELLOW]));
  while(key.ScanCode!=0x17) //ESC
  {
    GetKey(&key);
    if(key.ScanCode ==1 ) //UP
      draw_string(s_text1, 80, 100, &MyFontArray, &(gColorTable[DEEPBLUE]));
    else if(key.ScanCode ==2 ) //DOWN
      draw_string(s_text1, 80, 100, &MyFontArray, &(gColorTable[WHITE]));
    if(key.UnicodeChar == 0x31)
      SetLed(MyIoBaseAddr,LED1,LEDON);
    if(key.UnicodeChar == 0x32)
      SetLed(MyIoBaseAddr,LED2,LEDON);
    if(key.UnicodeChar == 0x33)
      SetLed(MyIoBaseAddr,LED1,LEDOFF);
    if(key.UnicodeChar == 0x34)
      SetLed(MyIoBaseAddr,LED2,LEDOFF);
  }
  SetMyMode(OldGraphicsMode);
  SwitchGraphicsMode(FALSE);
}
```

The logic of the code is not complicated. After setting the graphic mode and displaying the corresponding prompt characters, it directly enters the key acquisition loop. It will not exit the loop until the user's ESC key is received.

The up and down keys of the arrow keys are used to control the display and elimination of the string s_text1 on the screen; the number keys 1, 2, 3 and 4 are used to control the on and off of LED1 and LED2 respectively.

3 Testing

The compilation command is as follows:

```
C:\UEFIWorkspace>build -t VS2015x86 -p RobinPkg\RobinPkg.dsc -m RobinPkg\Drivers\YIE1CG\YIE1CG.inf -a X64
```

According to the method introduced in UEFI Development Exploration 75, flash YIE1CG.rom into the Flash of YIE001 and insert it into the actual machine for testing (you can also use the method introduced in UEFI Development Exploration 71 and use the load command for testing without flashing).

The interface of Option ROM is shown in Figure 1.

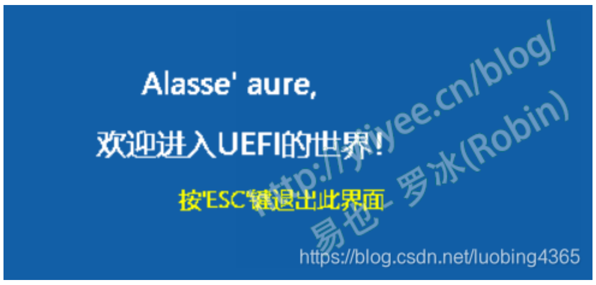
Alasse' aure,
欢迎进入UEFI的世界!
按'ESC'键退出此界面

Figure 1 The interface of YIE1CG

The demonstration of controlling the LED light by keyboard can only be done by recording the video and converting it into GIF. However, the GIF file is too large, so I won't post it. If you have the conditions, you can modify the code according to the method in this article to conduct the experiment.

Gitee address: https://gitee.com/luobing4365/uefi-explorer
The ROM file used in the project is located at: /76 YIE1CG