UEFI porting LVGL

原创 Anthony Posted on 2023-03-06 17:28:10 Read 1.7k Collection 5 Likes

Category Column: BIOS learning practice Article Tags: UEFI

Those who have assembled a game console by themselves should have seen it. When you enter the BIOS settings, a cool interface will appear in front of you. However, many BIOS still use the standard interface. Now there is a trend that Phoenix and Insyde are also slowly developing towards this GUI interface, and AMI's interface written in C++ is already very complete. I have always wanted to try it myself, and now it just happens.

Let's start with LVGL. Actually, I've always wanted to start with GUILite, so if there are any updates in this area later, it will basically be GUILite. LVGL is just a port, and I don't plan to study it in depth after the port. (LVGL is open source and can be used commercially for free. It seems that as long as you make some contributions to the open source, even if it's just writing porting documents)

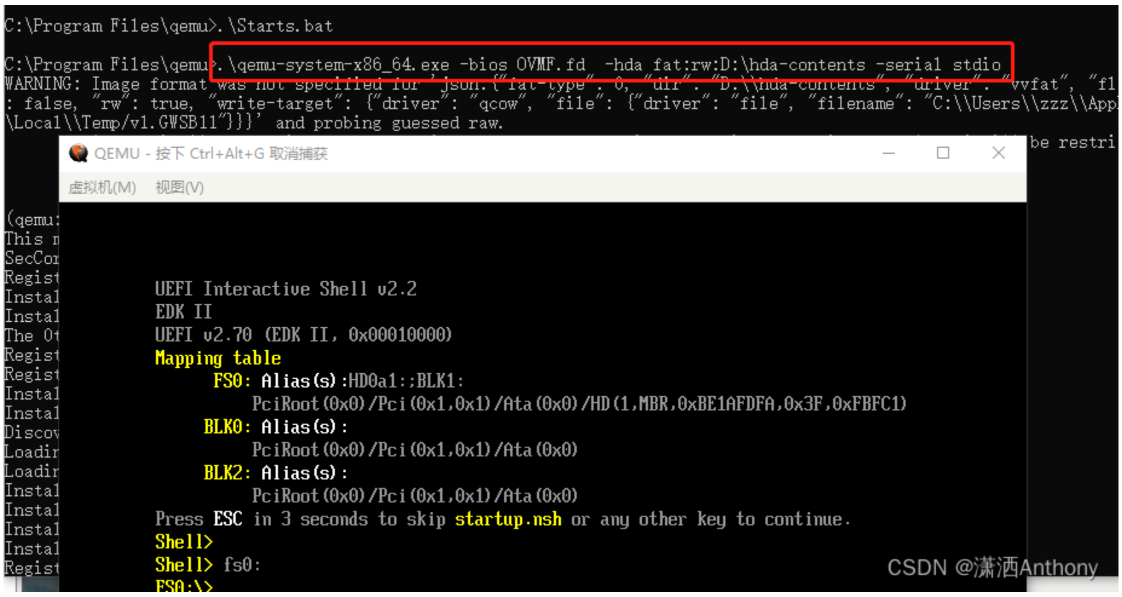
Aiying Blog - UEFI Development Learning 8 - Porting of LVGL GUI Library

You can take a look at this article. This article talks about a lot of things, and it is very well written. However, the input device mentioned at the end is a bit different, which makes it a little difficult to understand. During the porting process, I directly used the LVGL8.2 version for porting.

During the porting process, the main problems encountered were the compilation problems encountered when importing StdLib and the compilation problems encountered when importing LVGL. These problems have all been solved. I used OvmfPkg and compiled it with GCC/X64 in the Linux environment. After the compilation was completed, it ran through the Qemu emulator in Windows.

Regarding the compilation problem, I will upload the code temporarily through the network disk. The basic code is the code in my warehouse. You can compare the above two sets of codes. There are still many changes (Redfish content is imported into OvmfPkg, don't worry about it, just don't open the definition, otherwise it will not start, and the redfish code may not be able to run through Ovmf).

I will not repeat the content before inputting the device. The code and APP are also available. If you have a compiling environment, you can download it and try it yourself. First, let's implement the content of the interface:



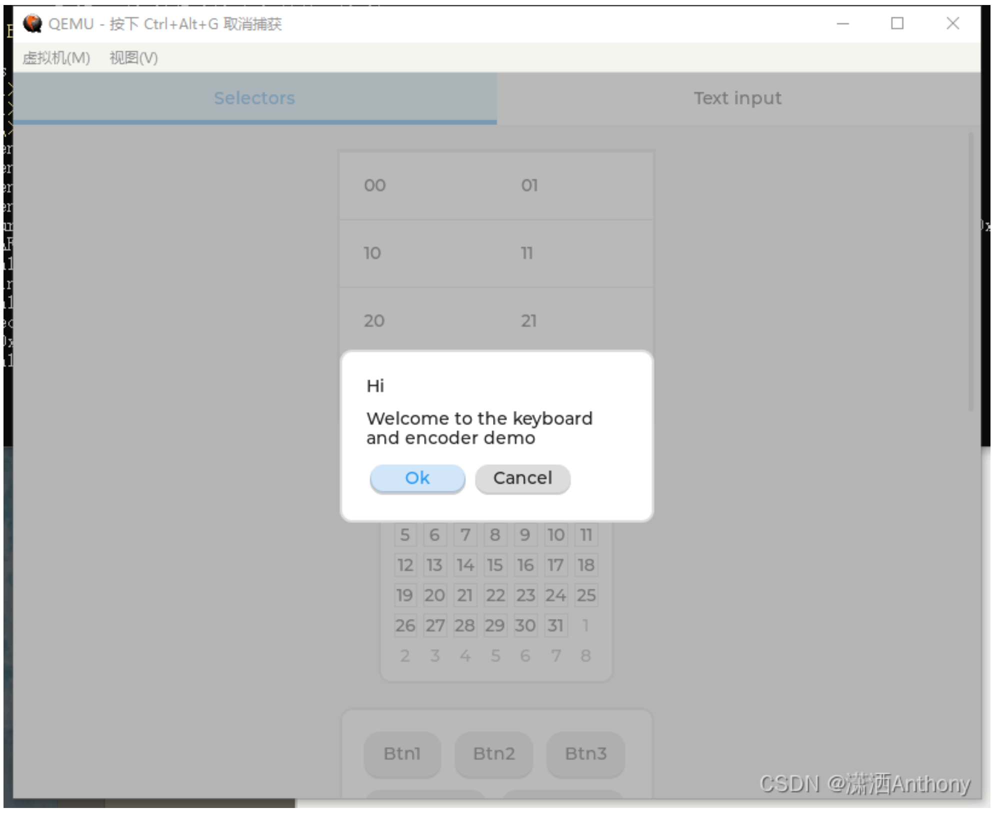
```
C:\Program Files\qemu>.\Starts.bat

C:\Program Files\qemu>.\qemu-system-x86_64.exe -bios OVMF.fd  -hda fat:rw:D:\hda-contents -serial stdio
WARNING: Image format was not specified for 'json:{"fat-type": 0, "dir": "D:\\hda-contents", "driver": "vvfat", "fl
: false, "rw": true, "write-target": {"driver": "qcow", "file": {"driver": "file", "filename": "C:\\Users\\zzz\\App
\Local\\Temp/v1.GWSB11"}}}' and probing guessed raw.
                                                                                                        be restri
(qemu:
This m
SecCor
Regist
Instal
Instal
The Ot
Regist
Regist
Instal
Instal
Discov
Loadir
Loadir
Instal
Instal
Instal
Instal
Regist
```

```
UEFI Interactive Shell v2.2
EDK II
UEFI v2.70 (EDK II, 0x00010000)
Mapping table
      FS0: Alias(s):HD0a1::BLK1:
          PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)/HD(1,MBR,0xBE1AFDFA,0x3F,0xFBFC1)
     BLK0: Alias(s):
          PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)
     BLK2: Alias(s):
          PciRoot(0x0)/Pci(0x1,0x1)/Ata(0x0)
Press ESC in 3 seconds to skip startup.nsh or any other key to continue.
Shell>
Shell> fs0:
FS0:\>
```

Here serial is turned on, DEBUG_ON_SERIAL_PORT in the code is to be turned on, enter fs0, open LvglTest.efi

You can see the interface:


QEMU - 按下 Ctrl+Alt+G 取消捕获
虚拟机(M)
视图(V)
Selectors
Text input
00
01
10
11
20
21
Hi
Welcome to the keyboard and encoder demo
Ok
Cancel
Btn1
Btn2
Btn3

The code of this interface is in the keypad_encoder under the demos of the Lvgl8.2 code. I put it directly into the C file. There is no problem with the porting of LVGL and the compilation can pass, but the key input has no response. If the key does not respond, this interface will lose a lot of fun. After my research, I found that the keyboard input still needs to be initialized.

GitHub - lvgl/lv_drivers at dev There is a keyboad.C and H file under indev. Take them out and put them under src/indev. The keyboard_init function in the keyboard.c file was originally empty. I added some code:

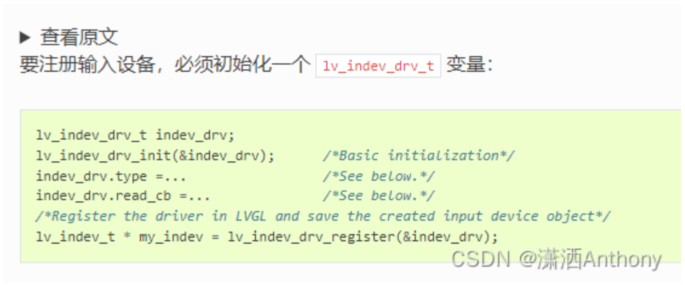
```
bool keyboard_read(lv_indev_drv_t * indev_drv, lv_indev_data_t * data);

void keyboard_init(void)
{
    lv_indev_drv_t indev_drv;
    lv_indev_drv_init(&indev_drv);      /*Basic initialization*/
    indev_drv.type =LV_INDEV_TYPE_KEYPAD ;               /*See below.*/
    indev_drv.read_cb =keyboard_read;
    lv_indev_t * my_indev = lv_indev_drv_register(&indev_drv);
}
```

The other functions are the same as described in the Aiying blog. This code is what I found in the document:

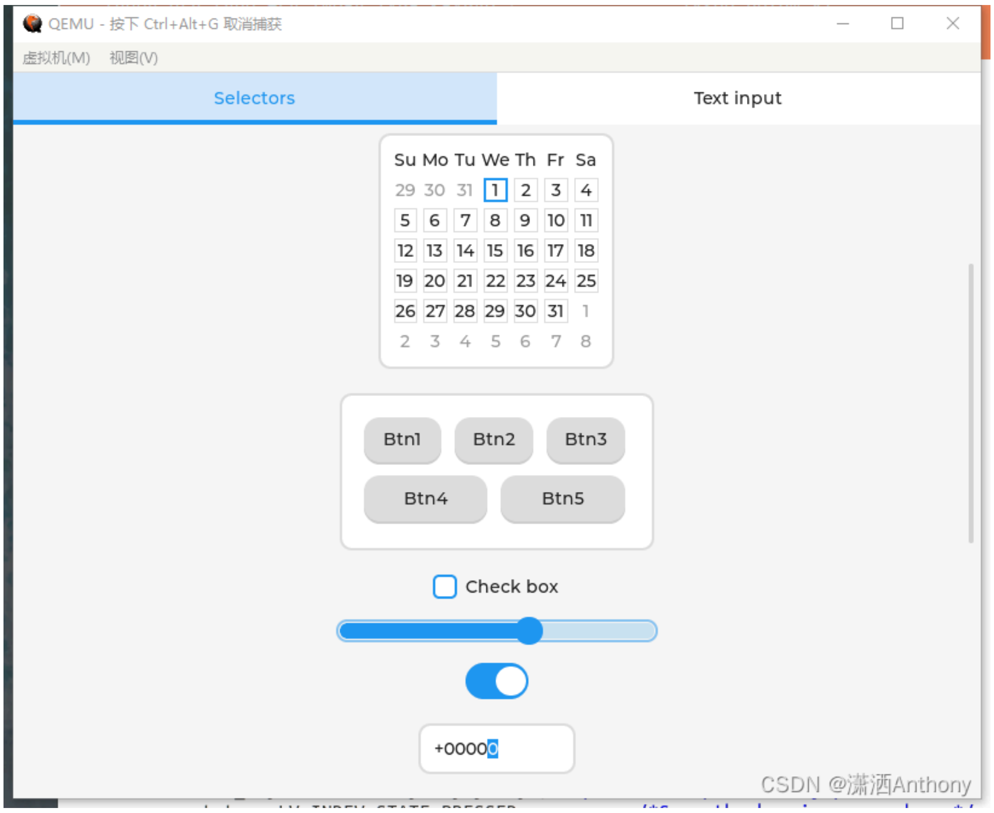
Input device interface — Baiwen.com LVGL Chinese Tutorial Documentation

Here's what he said:


▶ 查看原文
要注册输入设备，必须初始化一个 `lv_indev_drv_t` 变量：

```
lv_indev_drv_t indev_drv;
lv_indev_drv_init(&indev_drv);      /*Basic initialization*/
indev_drv.type =...                 /*See below.*/
indev_drv.read_cb =...              /*See below.*/
/*Register the driver in LVGL and save the created input device object*/
lv_indev_t * my_indev = lv_indev_drv_register(&indev_drv);
```

Finally, I called this keyboard_init function in the C file of the App, and the keyboard input problem was perfectly implemented:



As you can see, the interface is different from the previous one, because the interface can be operated normally, LVGL is transplanted, the interface elements can be used normally, and the input can be input normally, basically there is no big problem. Of course, there is also a mouse, the principle is the same, take a look at the documentation manual. After completing the above content, we can write some of our own interface programs, such as updating BIOS, you can set a cool interface, the whole cool progress bar, no problem at all.

The overall code:

Link: https://pan.baidu.com/s/1G6Ck342ns7w7RH9pp4qOTw
Extraction code: c79h