

# UEFI Development Exploration 65- YIE001PCIe Development Board (01 Introduction)

原创

luobing4365

Posted on 2021-01-05 13:33:47


Read 1.7k

Collection 4

Likes

copyright

Category Column: [UEFI Development](#) Article Tags: [uefi](#) [Low-level programming](#) [PCI-E](#) [Option ROM](#) [PCI-E Development Board](#)

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 haven't updated my blog for a while, mainly because I spent most of my time on company projects and finishing the manuscript. In the process of finishing the manuscript, when it came to the chapter about the PCIe protocol, I always felt that just introducing the protocol and UEFI code was a bit too boring.

Therefore, I overturned all the chapters I had written before, and changed the content of introducing the PCI protocol to implementing an actual UEFI Option ROM. In other words, I planned to make a PCIe development board with Flash ROM, inject the UEFI Oproam code into it, and plug the device into the computer to display the written interface, control the devices on the development board, and realize some interesting functions.

I named the development board YIE001, which comes from the abbreviation of our blog site. The current plan is as follows:

1. Implement basic Oproam code;
2. Implement interface and keyboard control functions;
3. Implement the marquee program;
4. Realize the key control function of the development board;
5. Realize the screen display function on the development board;
6. Port the basic GUI to the development board;
7. Make a game of snake.
8. More.....

The fifth one refers to the OLED screen of the I2C interface on the development board, not the host screen. I am not sure whether this function can be realized at present. The I2C timing of the PCIe chip Ch366 I use is very strange.

Another plan is about UEFI and USB. I plan to expand it a little bit and talk about how to use a single-chip microcomputer to achieve HID communication, how to access HID devices under Windows , and how to access HID devices under UEFI as a large topic. If there is enough time, I will also try to implement the method of accessing HID devices under the Linux system. This will explore all the knowledge that I think should be mastered from top to bottom of USB HID.

**This blog mainly introduces the hardware structure of the YIE001 that has just been proofed.**

The development board YIE001 is mainly made for developing UEFI Option ROM, and its purpose is to demonstrate how Option ROM code calls Protocol and how to control hardware . It is a PCIe board that provides toggle switch control, LED light control and I2C interface. Its product structure is shown in Figure 1.

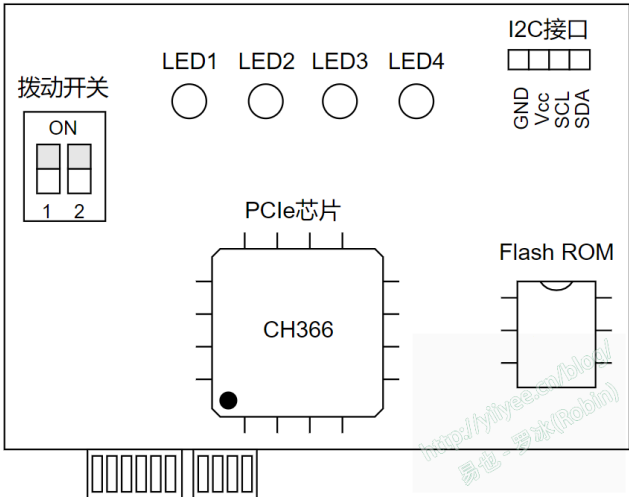


Figure 1 YIE001 development board structure

The actual board is shown in Figure 2.

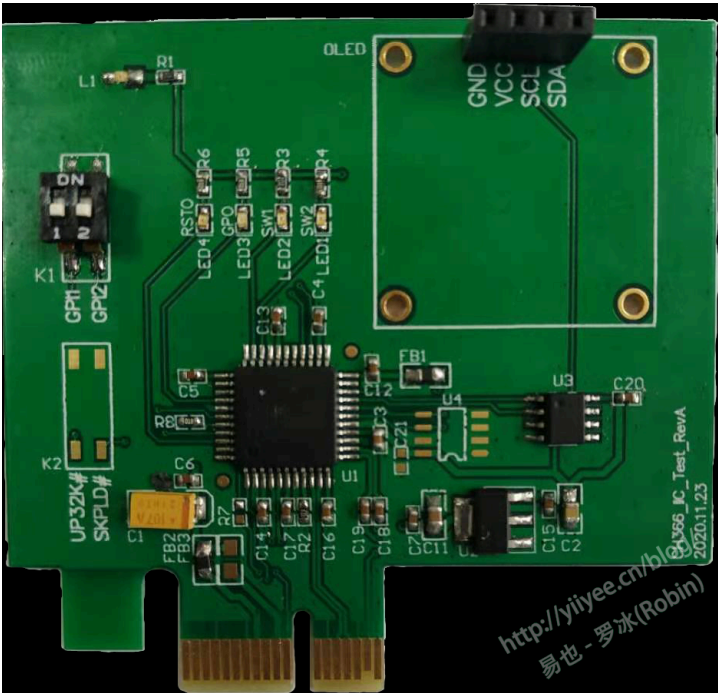


Figure 2 YIE001 development board

The main chip of YIE001 is CH366. When introducing the communication with PCI/PCIe devices in UEFI environment in Section 7.1.1, the functions of some of its internal registers were introduced. CH366 is a dedicated chip for the expansion ROM control card of PCI- Express bus. It supports 64KB to 1MB of electrically erasable read-only memory Flash ROM. The Flash ROM used in the development board is SST25VF010A from SST Company, with a capacity of 1Mb, that is, 128KB.

On YIE001, the hardware resources provided include 2 toggle switches, 4 LED lights, and an I2C interface. The CH366 on the development board uses a LQFP-44 lead-free package. GPO, and RSTO respectively; the I2C interface directly leads out the 4 I2C-compatible pins in the CH366 to facilitate the connection of other I2C devices.

Checking the chip manual of CH366, we know that LED1 and LED2 correspond to bits 0 and 1 of the control register of CH366. Setting the value of bit 0 of the control register to 0 can turn on LED1, and setting it to 1 can turn off LED1. The on and off of LED2 can also be achieved by setting the value of bit 1 of the control register. The correspondence of other hardware resources is shown in Figure 3.

寄存器名称	位址	属性	位的使用说明(默认值)	位值=0	位值=1
输出寄存器 GPOR (I/O 基址+00H 地址)	位 0	W	设定 SDA 引脚的输出值 (1)	低电平	高电平
	位 1	W	设定 SCL 引脚的输出值 (1)	低电平	高电平
	位 2	W*	设定 SCS 引脚的输出值 (1)	低电平	高电平
	位 3	W*	设置基本锁定状态 (0)	解除锁定	进入锁定
	位 4	W#	设置 SW 引脚的锁定状态 (0)	解除锁定	进入锁定
	位 5	W	使能支持被强制唤醒 (0)	不支持	支持
	位 6	W	设定 SDX 引脚的数据方向 (0)	输入	输出
	位 7	W	设定 SDX 引脚的输出值 (0)	低电平	高电平
控制寄存器 CTLR (I/O 基址+01H 地址)	位 0	W#	设置 SWO 引脚的输出 (0)	低电平	高电平
	位 1	W#	设置 SW1 引脚的输出 (1)	低电平	高电平
	位 2	W	预置复位后 SWO 引脚的输出 (0)	复位时加载到 0	
	位 3	W	预置复位后 SW1 引脚的输出 (1)	复位时加载到 1	
	位 4	WWW	软件可以读写的位变量 (000)	由应用程序定义	
	~位 6		不受 PCIe 总线复位的影响		
	位 7	W	复位时 SW 自动加载源选择 (0)	从外部配置芯片	从 CTLR 位 3 位 2
输入寄存器 GP1R (I/O 基址+02H 地址)	位 0	R	输入 SDA 引脚的状态 (1)	低电平	高电平
	位 1	R	输入 GP11 引脚的状态 (1)	低电平	高电平
	位 2	R	输入 GP12 引脚的状态 (1)	低电平	高电平
	位 3	R	输入 INT#引脚的状态 (1)	低电平	高电平
	位 4	R	输入 WAKIN#引脚的状态 (1)	低电平	高电平
	位 6	R	输入 UP32K#引脚的状态 (1)	低电平	高电平
	位 7	R	输入 SDX 引脚的状态 (1)	低电平	高电平
中断控制寄存器 INTCR (I/O 基址+03H 地址)	位 1	W*	全局中断使能 (0)	禁止中断	使能中断
	位 2	W*	INT#引脚中断输入的极性	低电平	高电平
辅助寄存器 AUXR (I/O 基址+18H 地址)	LED3 位 0	W	设定 GPO 引脚的输出值 (1)	低电平	高电平
	LED4 位 7	W	设定 RSTO 引脚的输出值 (1)	低电平	高电平

Figure 3 YIE001 hardware resource mapping (from CH366 chip manual)

This is the end of the introduction to the hardware structure of YIE001. In the next article, I will use this development board to implement various functions as planned and do some interesting experiments.

