BIOS Practice: Super IO-Smart Fan

原创 Anthony Modified on 2022-02-28 23:12:54 Read 9.2k Collection 23 Likes 2 Copyright CC 4.0 BY-SA
Category Column: BIOS learning practice Article Tags: Microcontroller stm32 Internet of Things

摘要 This article introduces the basic location and function of the super input and output chip (SIO), and elaborates on how to achieve intelligent control of computer fan speed (SmartFan) through the SIO chip, including key steps such as entering configuration mode, enabling the environmental controller, and obtaining the base address.

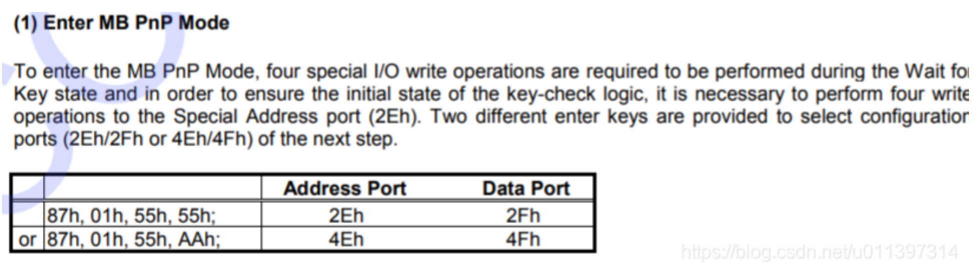
The summary is generated in C Know , supported by DeepSeek-R1 full version, go to experience>

The super input and output chip (SIO) is usually located at the lower left or upper left of the motherboard. The main chips used are Winbond and ITE, which provide control processing functions for the standard I/O interface on the motherboard.

Super means that it integrates the processing functions of PS/2 keyboard, PS/2 mouse, serial port COM, parallel port LPT interface, etc. These interfaces are slow I/O devices in the computer. They are all located on the right side of the rear of the motherboard. Its main functions include processing serial data transmitted from keyboard, mouse, serial interface and other devices, converting them into parallel data, and also responsible for the transmission and processing of parallel interface and floppy drive interface data.

The SIO chip is connected to the South Bridge through LPC, and the EC ROM of the desktop motherboard is mounted under SIO, so the initialization of LPC must include the initialization of SIO and EC. Of course, let's not talk about this now. Now we mainly talk about how to implement SIO initialization. Its initialization means the PS/2 keyboard, serial port, parallel port under SIO, as well as some LEDs, Beeps, fan control, etc. Let's take the smart fan as an example to see the operation process of SIO (ITE8738).

1. You can search for PnP Mode in Spec to find the base address. Address Port is usually 2E, Data Port is usually 2F. If you have a RW tool, you can enter IO Space, enter address 2E, and then enter (87h, 01h, 55h, 55h) in sequence at position 0x00 to enter MB PnP mode. You may find that every time you enter one, it will automatically change to 0xff. Don't think it is not effective. In fact, it has entered PnP Mode. For example, if you enter 87h and confirm it, it will immediately change back to 00. Don't panic, continue to enter 01h to confirm.



(1) Enter MB PnP Mode

To enter the MB PnP Mode, four special I/O write operations are required to be performed during the Wait fo Key state and in order to ensure the initial state of the key-check logic, it is necessary to perform four write operations to the Special Address port (2Eh). Two different enter keys are provided to select configuratior ports (2Eh/2Fh or 4Eh/4Fh) of the next step.

		Address Port	Data Port
	87h, 01h, 55h, 55h;	2Eh	2Fh
or	87h, 01h, 55h, AAh;	4Eh	4Fh

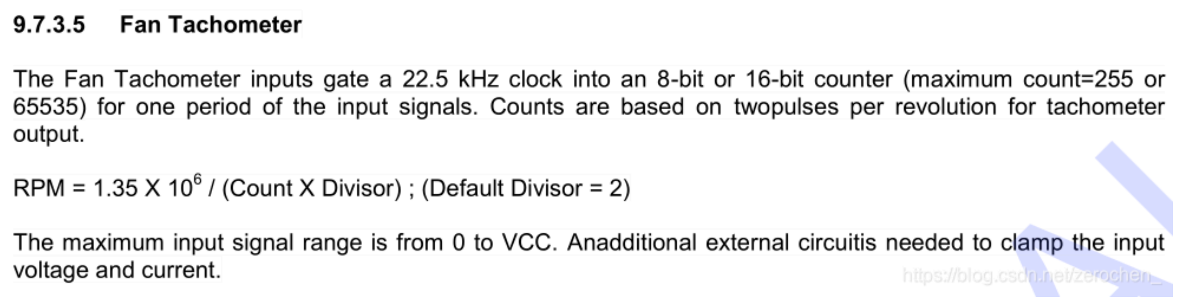
2. Enable EC (LDN=04h, Index 30h=01) By looking at SPEC, you can find that the logical device selection register is 07h, and its logical device number is 04h. At this time, enter IO Space, enter 2E, enter 07h in the control register 2E address, that is, 0x00, and then enter 04h in the data register 2F address, that is, 0x01, enter 30h in the control register 2E address, that is, 0x00, and enter 01h in the data register 2F address, that is, 0x01, to enable EC.

> Environment Controller Configuration Registers (LDN=04h)
> Environment Controller Activate (Index=30h, Default=00h)
> Bit Description:
> 7-1 Reserved
> 0 Environment Controller Enable
> 1: Enable
> 0: Disable
> This is a read/write register

3. Get EC Base Address. From SPEC, we can see that the base address needs to be read from registers 60h and 61h. 60h is the high eight bits, and 61h is the low eight bits, which together are the base address. Then the address port address of our EC controller is base+05h, and the data port address is base+06h.

The base address is determined by the logical device register (index=60h, 61h), address port (Base+05h), data port (Base+06h).

4. Smart Fan logic configuration.



9.7.3.5 Fan Tachometer

The Fan Tachometer inputs gate a 22.5 kHz clock into an 8-bit or 16-bit counter (maximum count=255 or 65535) for one period of the input signals. Counts are based on twopulses per revolution for tachometer output.

RPM = 1.35 X 10^6 / (Count X Divisor) ; (Default Divisor = 2)

The maximum input signal range is from 0 to VCC. Anadditional external circuitis needed to clamp the input voltage and current.

5. Exit MB PnP mode (index=02h, default=02h) This can also be seen in SPEC. The 02h part of the exit PnP Mode in the 2E/2F port is set to 01h.

At this point, fan control is complete.

Let's see how the following code is written (the library is not included). The function is not smartfan, it just turns the fan speed to the maximum.

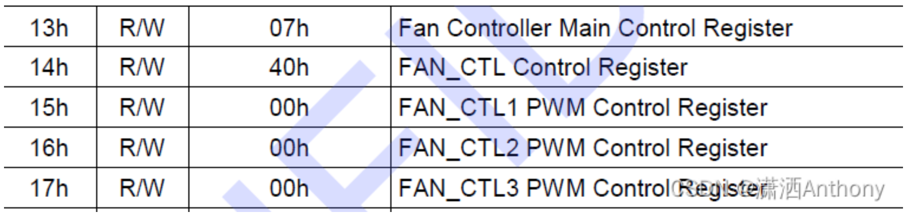
```
static unsigned char linux_inb(unsigned long port)
{
      return (*(volatile unsigned char *)(0x90000efdfc000000 + port));
}
#define linux_outb(port,val) \
do {\
      *(volatile unsigned char *)(0x90000efdfc000000 + (port)) = (val);  \
      }
while(0)

void EnterCOonfigMode(void)
{
  linux_outb(0x2E, 0x87);
  linux_outb(0x2E, 0x01);
  linux_outb(0x2E, 0x55);
  linux_outb(0x2E, 0x55);
  return;
}

void ExitConfigMode (void)
{
  UINT8  RegData = 0x00;
  linux_outb (0x2E, 0x02);
  RegData = linux_inb (0x2F) | BIT1;
  linux_outb (0x2E, 0x02);
  linux_outb (0x2F, RegData);
  return;
}

void SmartFanFunction(void)
{
 UINT16 addr;
 UINT8 base1,base2;
 UINT8 fandata1,fandata2;

 EnterConfigMode();

//LDN Selection
  linux_outb(0x2E,0x07);
  linux_outb(0x2F,0x04);

//Enable Environment Controller
  linux_outb(0x2E,0x30);
  linux_outb(0x2F,0x01);

//Get BaseAddress 0x290
  linux_outb(0x2E,0x60);
  base1=linux_inb(0x2F);
  DEBUG((EFI_D_INFO, "base1: 0x%x\n", base1));
  linux_outb(0x2E,0x61);
  base2=linux_inb(0x2F);
  DEBUG((EFI_D_INFO, "base2: 0x%x\n", base2));
  addr=(base1<<8&0xFF00)|base2;
  DEBUG((EFI_D_INFO, "GetFanBaseAddress: 0x%x\n", addr));

//Open ON/OFF mode
  linux_outb(addr+5,0x13);
  fandata1=linux_inb(addr+6);
  DEBUG((EFI_D_INFO, "fandata1: 0x%x\n", fandata1));
  linux_outb(addr+6,0x00);
  //Open ON control and FAN_CTL polarity high
  linux_outb(addr+5,0x14);
  fandata2=linux_inb(addr+6);
  DEBUG((EFI_D_INFO, "fandata2: 0x%x\n", fandata2));
  linux_outb(addr+6,0x87);

  ExitConfigMode();
}

void main()
{
SmartFanFunction();
return;
}
```

收起 ^

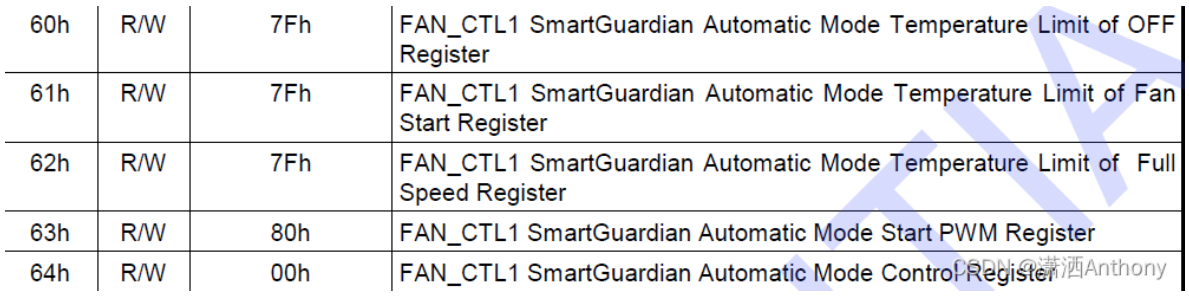
---------------February 28, 2022 22:26:05-----------------

Following the previous beep and LED, this time we are going to delve into the functional implementation of the Smart fan. First, we need to understand the operating mechanism of this smart fan, which monitors the temperature of the machine and then adjusts the fan speed according to the temperature. According to the document, we found Section 9.5, which is actually mostly IO space opened up for this smart fan.

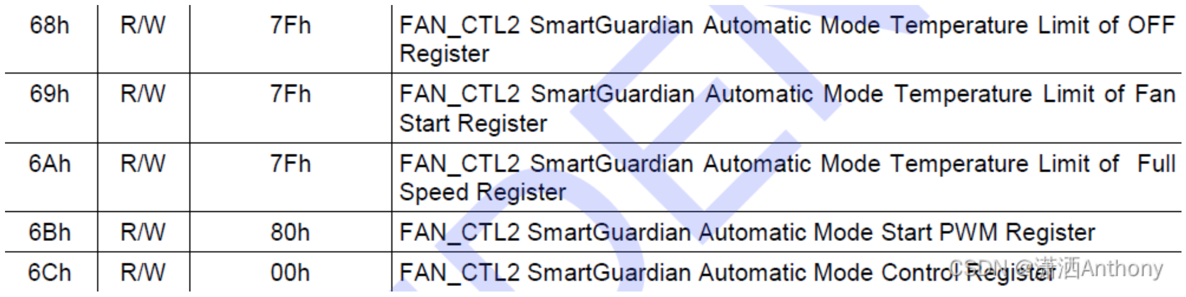
The following position is needed to operate the smart fan



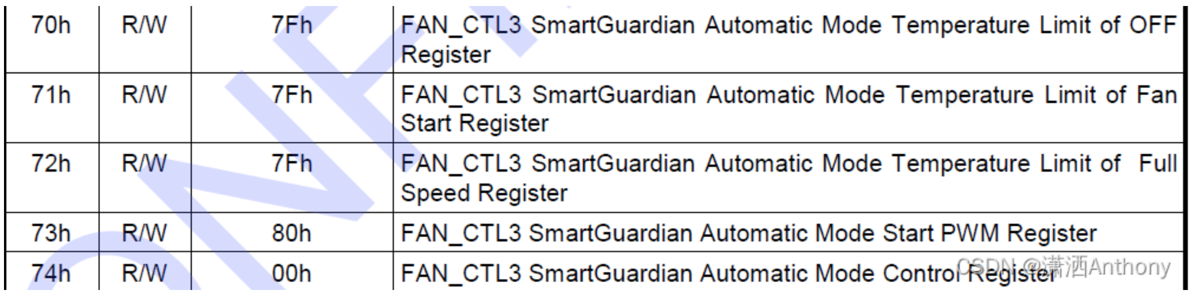
13h	R/W	07h	Fan Controller Main Control Register
14h	R/W	40h	FAN_CTL Control Register
15h	R/W	00h	FAN_CTL1 PWM Control Register
16h	R/W	00h	FAN_CTL2 PWM Control Register
17h	R/W	00h	FAN_CTL3 PWM Control Register



60h	R/W	7Fh	FAN_CTL1 SmartGuardian Automatic Mode Temperature Limit of OFF Register
61h	R/W	7Fh	FAN_CTL1 SmartGuardian Automatic Mode Temperature Limit of Fan Start Register
62h	R/W	7Fh	FAN_CTL1 SmartGuardian Automatic Mode Temperature Limit of Full Speed Register
63h	R/W	80h	FAN_CTL1 SmartGuardian Automatic Mode Start PWM Register
64h	R/W	00h	FAN_CTL1 SmartGuardian Automatic Mode Control Register

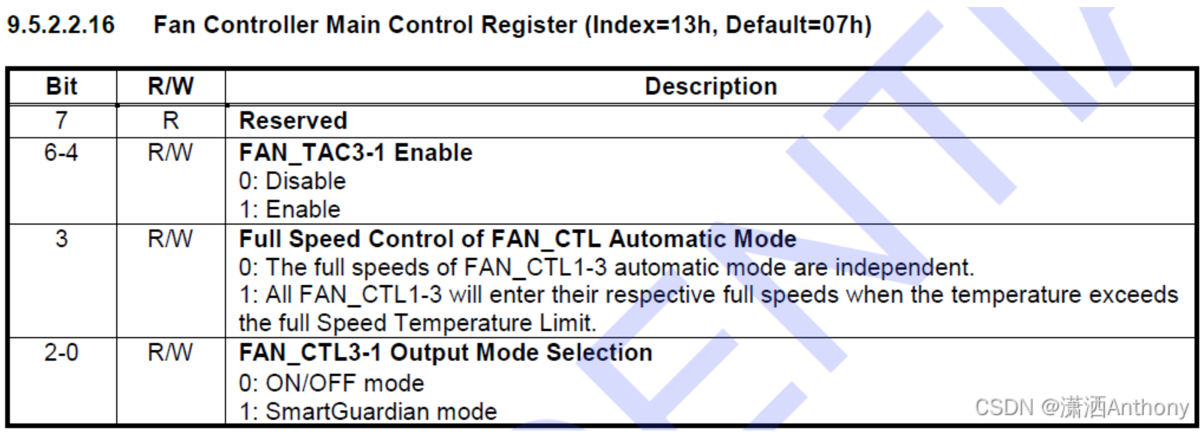


68h	R/W	7Fh	FAN_CTL2 SmartGuardian Automatic Mode Temperature Limit of OFF Register
69h	R/W	7Fh	FAN_CTL2 SmartGuardian Automatic Mode Temperature Limit of Fan Start Register
6Ah	R/W	7Fh	FAN_CTL2 SmartGuardian Automatic Mode Temperature Limit of Full Speed Register
6Bh	R/W	80h	FAN_CTL2 SmartGuardian Automatic Mode Start PWM Register
6Ch	R/W	00h	FAN_CTL2 SmartGuardian Automatic Mode Control Register



70h	R/W	7Fh	FAN_CTL3 SmartGuardian Automatic Mode Temperature Limit of OFF Register
71h	R/W	7Fh	FAN_CTL3 SmartGuardian Automatic Mode Temperature Limit of Fan Start Register
72h	R/W	7Fh	FAN_CTL3 SmartGuardian Automatic Mode Temperature Limit of Full Speed Register
73h	R/W	80h	FAN_CTL3 SmartGuardian Automatic Mode Start PWM Register
74h	R/W	00h	FAN_CTL3 SmartGuardian Automatic Mode Control Register

Each register has a corresponding description:



9.5.2.2.16 Fan Controller Main Control Register (Index=13h, Default=07h)

Bit	R/W	Description
7	R	**Reserved**
6-4	R/W	**FAN_TAC3-1 Enable** 0: Disable 1: Enable
3	R/W	**Full Speed Control of FAN_CTL Automatic Mode** 0: The full speeds of FAN_CTL1-3 automatic mode are independent. 1: All FAN_CTL1-3 will enter their respective full speeds when the temperature exceeds the full Speed Temperature Limit.
2-0	R/W	**FAN_CTL3-1 Output Mode Selection** 0: ON/OFF mode 1: SmartGuardian mode

Here 2-0 corresponds to 3-1, and 6-4 corresponds to 3-1. Don't get them mixed up. You can understand all the English. Just read them one by one. I don't want to record the specific operations. Anyway, just follow the steps and your own programming thinking.