BIOS Practice - HW Monitor

原创 Anthony Modified on 2022-03-01 14:04:17 Read 6.4k Collection 3 Likes

Category Column: BIOS learning practice Article Tags: UEFI

摘要 This article introduces how to use HWmonitor and IT8738 chips to monitor hardware status, including temperature sensors, fan speed, and IO space operations of the environmental controller (EC). It focuses on how to obtain the EC address, read and write temperature and fan data, and mentions the principles of temperature and speed calculation.

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

First, let's look at HW monitor, which is hardware monitoring, such as temperature, voltage, system fan, CPU fan, etc. In Section 9.5 of the IT8738 chip documentation, we can see that these are performed in the EC logic device built into the chip, which is environmental control (not embedded control):

9.5 Environment Controller

The Environment Controller (EC), built in the IT8738, includes seven voltage inputs, three temperature sensor inputs, four FAN Tachometer inputs, and four sets of advanced FAN Controllers. The EC monitors the hardware environment and implements the environmental control for personal computers.

The IT8738 contains an 8-bit ADC (Analog-to-Digital Converter), which is responsible for monitoring the voltages and temperatures. The ADC converts the analog inputs ranging from 0V to 2.8V (VREF) to 8-bit digital byte. With additional external components, the analog inputs can be made to monitor different voltage ranges, in addition to monitoring the fixed input range of 0V to 2.8V. Through external thermistors or thermal diodes, the temperature sensor inputs can be converted into 8-bit digital byte, enabling the sensor inputs to monitor the temperature of various components. A built-in ROM is also provided to adjust the non-linear characteristics of thermistors.

FAN Tachometer inputs are digital inputs with an acceptable range from 0V to 5V, and are responsible for measuring the FAN's Tachometer pulse periods.

The EC of the IT8738 provides multiple internal registers and an interrupt generator for programmers to monitor the environment and control the FANs. Both of the LPC Bus and Serial Bus interfaces are supported to accommodate the needs for various applications.

The environmental controller (EC) is built into the IT8738 and includes seven voltage inputs, three temperature sensor inputs, four fan tachometer inputs, and four sets of advanced fan controllers. The IT8738 contains an 8-bit ADC (analog-to-digital converter) responsible for monitoring voltage and temperature. The ADC converts analog inputs from 0V to 2.8V (VREF) into 8-bit digital bytes. With additional external components, the analog inputs can monitor different voltage ranges in addition to monitoring the fixed input range of 0 volts to 2.8 volts. Through an external thermistor or thermistor diode, the temperature sensor input can be converted into an 8-bit digital byte, allowing the sensor input to monitor the temperature of various components. A built-in ROM is also provided to adjust the characteristics of non-linear thermistors.

How to operate it has been mentioned in the basic introduction of smart fan before, I will just copy it below

9.5.1 Interface

LPC Bus: The Environment Controller of the IT8738 decodes two addresses.

Table 9-1. Address Map on LPC Bus

Register or Port	Address
Address register of EC	Base+05h
Data register of EC	Base+06h

Note 1: The Base Address is determined by the Logical Device configuration registers of the Environment Controller (LDN=04h, registers index=60h, 61h).

To access an EC register, the address of the register is written to the address port (Base+05h). Read or write data from or to that register via data port (Base+06h).

Take the LDN 4 EC device as an example. If we want to access its IO Space, we must first know its address. From the SPEC of IT8625, we can know that the address is the value in 0x60, 0x61.

The two values here cannot be used directly. According to the instructions, you need to add 05h and 06h to them respectively, and then access them using the index/data IO method. If the value of 0x60 is 02h and the value of 0x61 is 90h (these two values can be seen with RW), then:

index port address = 290h + 05h = 295h
data port address = 290h + 06h = 296h

Understanding the above, we can write the logic (PNP mode will not be written)

8.7 Environment Controller Configuration Registers (LDN=04h)

8.7.1 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.

I guess the above also needs to be enabled, so the total code should be as follows:


```
unsigned int GetEcAddress()
{
    unsigned int addr1,addr2,baseaddr;
    EnterPnPMode();

    IoWrite8(0x2E,0x07);
    IoWrite8(0x2F,0x04);

    IoWrite8(0x2E,0x30);
    IoWrite8(0x2F,0x01);

    IoWrite8(0x2E,0x60);
    addr1=IoRead8(0x2F);

    IoWrite8(0x2E,0x61);
    addr2=IoRead8(0x2F);

    ExitPnPMode();

    baseaddr=(addr1<<8&0xFF00) | addr2;

    return baseaddr;

}
```


OK, now we have the base address, with which we can read and write the IO space in EC:


```
unsigned int ReadEcSpace(unsigned int index){

    unsigned int data,Addr=0;
    unsigned int INDEX_PORT,DATA_PORT;

    Addr=GetEcAddress();
    INDEX_PORT = Addr+5;
    DATA_PORT = Addr+6;

    IoWrite8(INDEX_PORT,index);
    data=IoRead8(DATA_PORT);

    return data;
}

void WriteEcSpace(unsigned int index,unsigned int data){

    unsigned int Addr=0;
    unsigned int INDEX_PORT,DATA_PORT;

    Addr=GetEcAddress();
    INDEX_PORT = Addr+5;
    DATA_PORT = Addr+6;

    IoWrite8(INDEX_PORT,index);
    IoWrite8(DATA_PORT,data);

    return;
}
```

收起 ^

Now that the read and write functions are written, let's look at Section 9.5. This section introduces what the value of space is used for and what it represents:

Table 9-2. Environment Controller Registers

Index	R/W	Default	Registers or Action
00h	R/W	18h	Configuration Register
01h	R	00h	Interrupt Status Register 1
02h	R	00h	Interrupt Status Register 2
03h	R	00h	Interrupt Status Register 3
04h	R/W	00h	SMI# Mask Register 1
05h	R/W	00h	SMI# Mask Register 2
06h	R/W	00h	SMI# Mask Register 3
07h	R/W	00h	Interrupt Mask Interrupt Mask 1
08h	R/W	00h	Interrupt Mask Interrupt Mask 2
09h	R/W	80h	Interrupt Mask Interrupt Mask 3
0Ah	R/W	44h	Interface Selection Register
0Bh	R/W	0Fh	Fan PWM Smoothing Step Frequency Selection Register
0Ch	R/W	00h	Fan Tachometer 16-bit Counter Enable Register
0Dh	R	-	Fan Tachometer 1 Reading Register
0Eh	R	-	Fan Tachometer 2 Reading Register
0Fh	R	-	Fan Tachometer 3 Reading Register

In short, there is a series, let's take a look:

29h	R	-	TMPIN1 Temperature Reading Register
2Ah	R	-	TMPIN2 Temperature Reading Register
2Bh	R	-	TMPIN3 Temperature Reading Register
2Ch	R	-	TMPIN4 Temperature Reading Register

As mentioned in the previous article, the chip has three temperature sensor inputs, so we take the values of the first three, which is the temperature:


```
void TempRead()
{
    unsigned int temp0,temp1,temp2;

    temp0=ReadEcSpace(0x29);
    temp1=ReadEcSpace(0x2A);
    temp2=ReadEcSpace(0x2B);

    printf("Temperature0=%d",temp0);
    printf("Temperature1=%d",temp1);
    printf("Temperature2=%d",temp2);
    //这里只简单写下，温度肯定有范围，如果值为FF，那温度真是这样不得爆炸，所以要排除
}
```

收起 ^

The fan mainly has the following registers:

18h	R	-	Fan Tachometer 1 Extended Reading Register
19h	R	-	Fan Tachometer 2 Extended Reading Register
1Ah	R	-	Fan Tachometer 3 Extended Reading Register

0Dh	R	-	Fan Tachometer 1 Reading Register
0Eh	R	-	Fan Tachometer 2 Reading Register
0Fh	R	-	Fan Tachometer 3 Reading Register

Pay attention to the corresponding combination. For example, after reading the value of 18h, this value is shifted left by 8 bits, and then added to the value at 0Dh. Then, according to the fan speed formula, the fan speed is calculated.

Similarly, the voltage monitoring is 20h---28h. To read the values of these registers, it is estimated that calculations may be required, which may require hardware participation (guessing). OK, the overall hardware monitoring is learned here first-_-||