[19] 中华人民共和国国家知识产权局

[51] Int. Cl.
*G06F 11/22* (2006.01)

[12] 发明专利说明书

专利号 ZL 200410074679.5

[45] 授权公告日 2008 年 2 月 13 日

[11] 授权公告号 CN 100369006C

[22] 申请日 2004.9.13
[21] 申请号 200410074679.5
[73] 专利权人 英业达股份有限公司
地址 台湾省台北市
[72] 发明人 刘文涵 宋建福 刘 萍 刘一波 胡 幸
[56] 参考文献
CN2422674Y 2001.3.7
TW475151 2002.2.1
CN1273389A 2000.11.15
US6185677B1 2001.2.6
审查员 钟文芳

[74] 专利代理机构 北京律诚同业知识产权代理有限公司
代理人 梁 挥 祁建国

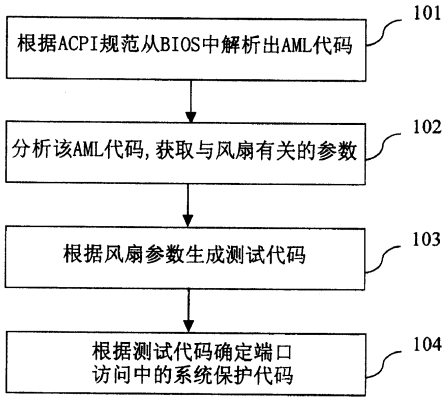
权利要求书 1 页 说明书 13 页 附图 2 页

[54] 发明名称

# 一种计算机风扇参数测试码的生成方法

[57] 摘要

本发明涉及一种计算机风扇参数测试码的生成方法，利用 ACPI 规范从 BIOS 中解析出与风扇有关的参数，以有效获取并设置风扇测试代码，该方法包括：根据 ACPI 规范读取内存中的 BIOS 映象文件，并解析出 AML 代码；找出该 AML 代码中与风扇有关的关键词，并将该些关键词由 AML 码转译为 ASL 码，以获取与风扇有关的参数；根据与风扇有关的参数生成测试代码；及根据该测试代码确定端口执行中受到系统保护的程序。 本发明可自动获取与风扇相关的参数，开发效率高，生成的测试代码稳定性高，通用性好。



1、一种计算机风扇参数测试码的生成方法，利用 ACPI 规范从 BIOS 中解析出与风扇有关的参数，以有效获取并设置风扇测试代码，其特征在于，该方法包括：

根据 ACPI 规范读取内存中的 BIOS 映象文件，并解析出 AML 代码；

找出该 AML 代码中与风扇有关的关键词，并将该些关键词由 AML 码转译为 ASL 码，以获取与风扇有关的参数；

根据与风扇有关的参数生成测试代码；及

根据该测试代码确定端口执行中受到系统保护的程序。

2、根据权利要求 1 所述的计算机风扇参数测试码的生成方法，其特征在于，该与风扇有关的参数包括风扇的硬件标识符、风扇状态、风扇状态的设定方法、选择测试风扇的途径。

3、根据权利要求 2 所述的计算机风扇参数测试码的生成方法，其特征在于，该风扇状态包括风扇停、慢、快状态。

4、根据权利要求 2 所述的计算机风扇参数测试码的生成方法，其特征在于，该测试风扇的途径包括 SystemIO、EmbeddedController、SMBus、CMOS。

5、根据权利要求 1 所述的计算机风扇参数测试码的生成方法，其特征在于，该生成测试代码的步骤中，包括设定风扇的转速。

6、根据权利要求 1 所述的计算机风扇参数测试码的生成方法，其特征在于，该生成测试代码的步骤，是生成 C++测试代码。

# 一种计算机风扇参数测试码的生成方法

## 技术领域

本发明涉及一种计算机硬件参数的获取与测试码生成方法，特别是涉及一种计算机风扇参数测试码的生成方法。

## 背景技术

随着计算机的速度越来越快，发热量也越来越大，在这方面，以 CPU（中央处理器，central processing unit）更为突出，因为 CPU 的集成度高达几百万晶体管，所以发热量之大让人难以想象。普通的 CPU 表面温度甚至可以达到 40-60 度，而 CPU 内部则更是高达 80 度甚至于上百度。如果计算机经常发生死机、蓝屏错误、IE 错误、打开程序错误、丢失数据、自动重启等问题，不是操作系统有问题，就是可能因为 CPU 过热而造成的。

如今，风扇在 CPU 分冷系统中发挥着决定性的作用，由于 CPU 的发热量实在是太大，只靠散热片的作用是不够的，因此一定要风扇来帮忙。

另一方面，每一台计算机，特别是笔记本计算机，在出厂之前为保证质量都要对所有硬件进行严格的测试。由于 CPU 风扇是确保 CPU 工作在安全温度范围的重要部件，对风扇的品质要求也越来越高，因此风扇测试也是计算机硬件测试的重要组成部分。

对于测试方来说，对风扇的测试目前都是通过在测试程序中先设置风扇转速，然后再读取风扇转速，通过判断两者是否一致，来实现风扇的测试。

然而，通常情况下，每个计算机厂商控制 CPU 风扇的方法都是不同的，甚至同一厂商的不同系列的机型，其控制方法也是不同的。以前，新机型风扇测试程序的开发都要经过以下步骤：

1、查找新机型的 BIOS 开发小组；

2、向 BIOS 小组询问新机型的系统架构、有关风扇的参数和设置方法；

3、根据获得的相关参数编制程序；

4、利用新机型的样机测试编制好的程序；

5、如有问题仍要重复以上的 2-4；

6、将测试成功的方法添加到测试程序中。

以上方法存在开发周期长，兼容性差，程序维护成本高且稳定性很差。主要原因就在于 BIOS 小组提供的风扇参数只是针对风扇的底层实现方法而言的，而真正的测试都是在操作系统层下完成。这样一来，如果只是直接对风扇操作会出现与操作系统发出的风扇指令冲突的现象，轻则导致程序中断，严重的会造成死机等情况。

而另一方面，关于计算机的电源管理，现在有 APM(Advanced Power Management，高级电源管理)和 ACPI(Advanced Configuration Power Interface 高级配置电源界面)两种规范。特别是 ACPI，它是一项电源管理的新技术，由操作系统可以直接和使用者直接互动，即当操作系统察觉现在某部分功能没有使用，便会自动降低其功能来减少电能使用，以储备更多电能，ACPI 可以被运用在不同程序语言的操作系统，例如 windows 2000 对笔记本计算机电池的运作，可以通过电源选项来控制，从电池计量表得知电池使用量等。如果要使用 ACPI 则必须从硬件、BIOS、及驱动程序均要符合，则 ACPI 操作系统才能正常地运作。

ACPI 标准定义了硬件寄存器、BIOS 接口(包含配置表格、控制方法以及主板设备列举和配置)、系统和设备的电源状态和 ACPI 热模型。BIOS 提供的支持代码不是用汇编语言，而是用 AML(ACPI Machine Language，ACPI 机器码，以二进制方式存在，为了使用方便通常以十六进制表示)编写的。

尽管 ACPI 是用于电源管理的新技术，但其中也包含有丰富的硬件信息，其中是否有风扇参数的描述，如何获取并利用这些参数编制风扇测试程序，已成为业界急需解决的问题。

## 发明内容

本发明所要解决的技术问题在于提供一种计算机风扇参数测试码的生成方法，可自动获取与风扇有关的参数，提高开发效率与稳定性。

为了实现上述目的，本发明提供了一种计算机风扇参数测试码的生成方法，利用 ACPI 规范从 BIOS 中解析出与风扇有关的参数，以有效获取并设置风扇测试代码，其特点在于，该方法包括：根据 ACPI 规范读取内存中的 BIOS

映象文件，并解析出 AML 代码；找出该 AML 代码中与风扇有关的关键词，并将该些关键词由 AML 码转译为 ASL 码，以获取与风扇有关的参数；根据与风扇有关的参数生成测试代码；及根据该测试代码确定端口执行中受到系统保护的程序。

上述计算机风扇参数测试码的生成方法，其特点在于，该与风扇有关的参数包括风扇的硬件标识符、风扇状态、风扇状态的设定方法、选择测试风扇的途径。

上述计算机风扇参数测试码的生成方法，其特点在于，该风扇状态包括风扇停、慢、快等状态。

上述计算机风扇参数测试码的生成方法，其特点在于，该测试风扇的途径包括 SystemIO、EmbeddedController、SMBus、CMOS 等。

上述计算机风扇参数测试码的生成方法，其特点在于，该生成测试代码的步骤中，包括设定风扇的转速。

上述计算机风扇参数测试码的生成方法，其特点在于，该生成测试代码的步骤，是生成 C++测试代码。

本发明功效，在于可自动获取与风扇有关的参数，开发效率高，生成的测试代码稳定性高，通用性好。

以下结合附图和具体实施例对本发明进行详细描述，但不作为对本发明的限定。

## 附图说明

图 1 为本发明的计算机风扇参数测试码的生成方法流程图；

图 2 为本发明的二进制代码翻译成 ASL 代码的流程图。

其中，附图标记：

步骤 101-根据 ACPI 规范从 BIOS 中解析出 AML 代码

步骤 102-分析该 AML 代码，获取与风扇有关的参数

步骤 103-根据风扇参数生成测试代码

步骤 104-根据测试代码确定端口访问中的系统保护代码

步骤 201-查找关键词“41 D0 0C 0B”

步骤 202-判断是否找到

步骤 203-解析并翻译二进制代码为 ASL 码得到相应 Power Resource 对象名

步骤 204-查找关键词“5B 84 XX YY”

步骤 205-解析并翻译二进制代码为 ASL 码，得到相应 Method 对象名

步骤 206-查找关键词“14 XX YY”

步骤 207-翻译并解析二进制代码为 AML 码，得到相应 Method 对象名

## 具体实施方式

现在的绝大部分操作系统都已支持 ACPI，ACPI 是由一些表状结构和一些二进制数码组成，这些表状结构和二进制代码被固化到计算机的 BIOS 中，在操作系统启动之前被调入内存，其中一部分是操作系统不可以更改的。就在这部分固定不变的内容中，包含了许多硬件的端口地址、操作偏移量以及硬件的操作数。

在这些硬件信息里，本发明可以从中自动查找获取与风扇测试相关的内容。这些内容是以二进制代码(即 AML 码)的形式记录的，根据 ACPI 的规范，可以将其翻译成一种语言即 ASL(ACPI Source Language)码，但是 ASL 码不能够直接运行，在这些早已经记录在 BIOS 的语言中，本发明可以解析出设置/读取风扇转速的途径和硬件端口以及相应的操作参数。而这些信息正是编制风扇测试程序必不可少的。

基于上述原因，本发明提供了一种新的方法，即通过分析 ACPI 中描述的有关读取/设置风扇转速部分二进制代码得到风扇测试码的生成方法。

请参阅图 1，为本发明的计算机风扇参数测试码的生成方法流程图，首先步骤 101，根据 ACPI 规范从 BIOS 中解析出 AML 代码；步骤 102，分析该 AML 代码，获取与风扇有关的参数；步骤 103，根据风扇参数生成测试代码；步骤 104，最后根据测试代码确定端口访问中的系统保护代码。

其中，该根据 ACPI 规范从 BIOS 中解析出 AML 代码的步骤，是通过读取内

存中的BIOS图像而获取的，首先在BIOS中搜寻与风扇信息有关的关键词，然后再将该关键词从二进制代码(AML码)翻译为ASL代码。

与风扇有关的参数包括风扇的硬件标识符、风扇状态、风扇状态的设定方法、风扇的测试实现途径等。其中，风扇状态包括风扇停、慢、快等状态，风扇测试实现途径包括SystemIO、EmbeddedController、SMBus、CMOS等。

生成的测试代码可以是C++代码。生成测试代码后，还可以根据具体情况分析，进一步精炼测试代码。

下面以一具体实施例来说明本发明。首先需要读取内存中的BIOS映像(AML)找到关键词“41 D0 0C 0B”(对应的ASL码：PNP0C0B)，通常会找到几组关键词组合，每一组都代表一种风扇状态信息，如：风扇停、慢、快等状态。每一种状态信息由_HID、_UID以及_PR0等信息组成。PNP0C0B就是风扇的_HID(Hardware ID)，正是利用该标识找到了风扇设备，_HID关键词为“08 5F 48 49 44”。_UID(Unique persistent ID)关键词为“08 5F 55 49 44”，描述了风扇状态的序号。每一组状态有一个相对唯一的ID。_PR0(Power Resource for D0)关键词为“08 5F 50 52 30”，描述了此状态的设定方法。跟在关键词后的第一个字节（Byte）是操作码，操作码是信息格式的起始关键词。即不同的操作码对应不同的信息格式。以AML码“12”为例，所代表的ASL操作码为“Package”，相应第二字节是“Package Length”即Package信息的位组长度。第三位字节是“Package Lead Byte”，注意这部分很可能不只占一个字节，具体结构描述为：

bit 7-6：第一个字节的高两位表示了随后的Package Length的字节数，如果Package Length只有一个字节即此两位值均为零则bit 0-5表示Package Length；

bit 5-4:预留；

bit 3-0:标识Package长度的最末字节。

第三字节后就是_PR0的详细信息。此部分通常只是列出了有关状态实现的对象名。(为了方便后面的论述假设一个对象名为”F001”)根据这个对象名进一步查找风扇测试的实现途径。

按照对象名“F001”在内存中的BIOS图像找关键词“5B 84 XX F0 01”。此关键词分三部分，第一部分“5B 84”是“PowerResOP”对象编码名(Object

Encoding Name)；第二部分“XX” 是“Package Lead Byte”，标识了该对象描述部分的长度；第三部分“F0 01”就是准备描述的对象名称。关键词之后是“PowerResOP”的参数部分，一个字节表示了该对象的 systemlevel 参数；再后面两个字节是 resourceorder 参数。至此有关“PowerResOP”对象的描述结束。在对象描述部分的后面是有对象相关方法的描述。该对象通常包括三种方法：_STA、_ON、_OFF，分别为得到风扇状态、打开该状态、关上该状态。在此仅以“_ON”为例说明，其它方法与此相仿。

方法(MethodOP)对象的二进制标识编码是“14”。然后是此方法描述部分的长度占一个字节。然后是方法参数。方法有两个参数，第一个参数是名称串“_ON_”，第二个参数占一个字节其中每一个 bit 表示如下：bit 0-2：参数个数(0--7)，bit 3：SerializeFlag，值“0”表示 NotSerialized，值“1”表示 Serialized，bit 4-7：SyncLevel(0x00-0x0f)。Serialize 规则用来防止方法的重复调用，如果创建 namespace 对象，这个规则尤其重要，没有这个规则，当尝试创建同一个 namespace 对象重复调用方法将失败。SyncLevel 参数声明同步对象的逻辑嵌套的层次，范围是 0-15。再后面通常是另一个方法对象(假设此方法对象名为“F002”)，如果此对象有参数，参数前会有参数标识 0x0a、0x0b、0x0c、0x0e、0x0d 分别表示参数的数据类型为 byte、WORD、DWORD、QWORD、String。至此对象 PowerResOP 翻译完毕。下面就是追踪方法对象“F002”。

按照对象名“F001”在内存中的 BIOS 图像找关键词“14 XX F0 02”。第一个字节是上面讲过的方法(Method)对象的二进制标识编码是“14”。第二个字节“XX”是 “Package Lead Byte”，因此这部分很可能不只占一个字节。后面是该方法对象的具体实现。因为不同机型的具体实现方法不同，但其翻译过程与上面的论述相仿，这里不一一介绍了。

请参阅图 2，简要说明了上述由二进制代码翻译成 ASL（ACPI Source Language，ACPI 源语言）代码的过程，首先步骤 201，查找关键词“41 D0 0C 0B”，步骤 202，判断是否找到，步骤 203，如果找到，则解析并翻译二进制代码为 ASL 码得到相应 Power Resource 对象名，步骤 204，接着查找关键词“5B 84 XX YY”，其中，XX：“Package Length”，YY：“Power Resource 对象名”，步骤 205，然后解析并翻译二进制代码(AML 码)为 ASL 码，得到相

应 Method 对象名，步骤 206，然后查找关键词“14 XX YY”，其中，XX：“Package Length”，YY：“Method 对象名，步骤 207，翻译并解析二进制代码为 ASL 码，得到相应 Method 对象名。

以 HP 系列笔记本计算机的 Ruby 机型为例，ACPI BIOS 有关风扇部分二进制编码翻译成 AML 码的对照描述如下：

1、搜索关键词“41 D0 0C 0B”：

```
41 D0 0C 0B 08 5F 55 49 44 0A 00        A....._UID..
08 5F 50 52 30 12 06 01 43 32 30 36 5B 82 22 43
._PR0...C206[.”C
32 30 42 08 5F 48 49 44 0C                    20B._HID.
Name(_HID  (Path \_TZ_.C20A._HID),EisaId(“PNP0C0B”))
Name(_UID  (Path \_TZ_.C20A._UID),0x00)
Name(_PR0  (Path \_TZ_.C20A._PR0),Package(0x01)
{
C206
})
```

2、搜索关键词“5B 84 XX YY”

```
5B 84 36 43 32 30 36 00 00.    rh..a.a[.6C206..
00 14 0F 5F 53 54 41 00 A4 43 31 46 46 0A 01 0A
..._STA..C1FF...
80 14 0E 5F 4F 4E 5F 00 43 32 30 30 0A 01 0A 80
..._ON_.C200....
14 0E 5F 4F 46 46 00 43 32 30 31 0A 01 0A 9E
.._OFF.C201....
PowerResource (C206  (Path \_TZ_.C206), 0x00, 0x0000)
{
Method (_STA  (Path \_TZ_.C206._STA), 0, NotSerialized)
{
Return (C1FF (0x01, 0x80))
}
```

```
Method (_ON  (Path \_TZ_.C206._ON_), 0, NotSerialized)
{
C200 (0x01, 0x80)
}
Method (_OFF  (Path \_TZ_.C206._OFF), 0, NotSerialized)
{
C201 (0x01, 0x9E)
}
}
```

3、搜索关键词”14 XX C200”

```
14 4B 08 43 32 30 30 0A 79 68 0A 01 60        .K.C200.yh..`
76 60 70 0A 9D 5C 2F 05 5F 53 42 5F 43 30 34 36
v`p..\._SB_C046
43 30 35 39 43 30 45 39 43 31 31 45 70 0A 62 5C
C059C0E9C11Ep.b\
2F 05 5F 53 42 5F 43 30 34 36 43 30 35 39 43 30
._SB_C046C059C0
45 39 43 31 31 46 A0 3E 93 7B 43 31 46 45 60 00
E9C11F.>.{C1FE`.
0A 00 70 0A 92 5C 2F 05 5F 53 42 5F 43 30 34 36
..p..\._SB_C046
43 30 35 39 43 30 45 39 43 31 31 45 70 69 5C 2F
C059C0E9C11Epi\.
05 5F 53 42 5F 43 30 34 36 43 30 35 39 43 30 45
._SB_C046C059C0E
39 43 31 31 46 7D 43 31 46 45 68 43 31 46 45
9C11F}C1FEhC1FE
Method (C200  (Path \_TZ_.C200), 2, Serialized)
{
ShiftLeft (Arg0, 0x01, Local0)
```

```
Decrement (Local0)
Store (0x9D, \_SB.C046.C059.C0E9.C11E)
Store (0x62, \_SB.C046.C059.C0E9.C11F)
If (LEqual (And (C1FE, Local0), 0x00))
{
Store (0x92, \_SB.C046.C059.C0E9.C11E)
Store (Arg1, \_SB.C046.C059.C0E9.C11F)
}
Or (C1FE, Arg0, C1FE)
}
```

4、搜索关键词”C1 1E” “C1 1F”

```
5B 80 43 31 31 44          [.C11D
01 0A 3E 0A 02 5B 81 10 43 31 31 44 01 43 31 31
..>..[..C11D.C11
45 08 43 31 31 46 08                    E.C11F.
OperationRegion (C11D, (Path \_SB_.C046.C059.C0E9.C11D),
SystemIO, 0x3E, 0x02)
Field (C11D, ByteAcc, NoLock, Preserve)
{
C11E,   8,
C11F,   8
}
```

然后，由 ASL 代码可以得到不同机型测试风扇的途径。实现风扇测试有多种途径，常用到的是 SystemIO 访问。其它方法如 EmbeddedController，SMBus，CMOS 访问等因与 SystemIO 相仿且用到的较少这里不作详细说明。通常在得到的 ASL 中搜索标识符 OperationRegion，该语句的第二个参数标明了风扇测试的途径。该语句的基本格式为：

```
OperationRegion(
RegionName, //区域名称
RegionSpace, //区域空间关键词，就是风扇测试实现的途径
```

Offset, //相对区域空间基址偏移量，整数型

Length //区域长度，整数型

)

以本文实例为例分析测试方法的获取过程。

在实例 ASL 中搜索“OperationRegion”可以得实例第 4 步代码如下：

OperationRegion (C11D (Path _SB_.C046.C059.C0E9.C11D), SystemIO, 0x3E, 0x02)

根据语句基本格式可以解释为：操作区域“C11D”属于 SMBus 设备，通过 System IO 区域空间偏移 0x3E 字节访问，该区域占用 System IO 两个字节即 0x3E，0x3F。

另举通过 EmbeddedControl 访问测试风扇的实例：

OperationRegion(ECF2(Path_SB_.PCI0.LPCB.EC0_.ECF2), EmbeddedControl, 0x00, 0xFF)

表示操作区域“ECF2”属于 SMBus 设备，通过 EmbeddedControl 区域空间偏移 0x00 字节访问，该区域占用 EmbeddedControl 空间 256 个字节。

紧跟在 OperationRegion 语句之后的是 Field 语句。该语句是对应操作区域的具体描述，声明了一系列的已命名的数据对象的值以及排列次序。基本格式如下：

Field(

RegionName, //区域名称=>OperationRegion

AccessType, //访问类型

LockRule, //加锁规则

UpdateRule //更新规则

) {FieldUnitList}

针对上面的实例：

Field (C11D, ByteAcc, NoLock, Preserve)

{

C11E, 8,

C11F, 8

}

解释为：区域对象名“C11D”，“ByteAcc”表示访问方式为字节，“NoLock”表示访问域内对象时不关心全局锁(Global Lock 后面将有说明)，“Preserve”预留。“C11E”位于对象“C11D”所在区域的第一个字节，因为占 8Bit。同样对象“C11F”也占一个字节。

利用解析 ASL 码获取风扇相关参数及控制方法，即可生成相应 C++代码：

```
ASL:  Method (C200  (Path \_TZ_.C200), 2, Serialized)
C++:  void SetFanOn(unsigned char arg0, unsigned char arg1)
{
ASL: ShiftLeft (Arg0, 0x01, Local0)
C++:  unsigned char Local0;
C++:  Local0 = arg0<<1;

ASL: Decrement (Local0)
C++:  Local0--;

ASL: Store (0x9D, \_SB.C046.C059.C0E9.C11E)
C++:  outportb(0x3E, 0x9D);

ASL: Store (0x62, \_SB.C046.C059.C0E9.C11F)
C++:  outportb(0x3F, 0x62);

ASL: If (LEqual (And (C1FE, Local0), 0x00))
C++:  unsigned char flag = 0;
C++:  if ((flag & Local0) == 0)
{
ASL: Store (0x92, \_SB.C046.C059.C0E9.C11E)
C++:    outportb(0x3E, 0x92);

ASL: Store (Arg1, \_SB.C046.C059.C0E9.C11F)
C++:  outportb(0x3F, Arg1);
```

```
}
ASL: Or (C1FE, Arg0, C1FE)
C++:  flag = flag | arg0;
}
```

上面的实例变量 flag 是 ACPI 系统中标志风扇状态的字节，在设置风扇状态时可以忽略。参数 arg0 代表状态代码，在设置风扇转速时也可以忽略。arg1 代表需要打开的风扇转速值。精炼后的设置风扇转速 C++代码为

```
void SetFanOn(unsigned char speed)
{
    outportb(0x3E, 0x92);
    outportb(0x3F, speed);
}
```

对于确定系统保护码，可以通过分析 OperationRegion 语句中对象的 LockRule 参数的描述确定是否有必要在设置风扇转速时获得“Global Lock”。“Global Lock”用来在 OSPM(Operation System Power Management)环境和外部控制器环境(如 SMI)之间同步访问共享硬件资源。在某一个时刻，这个 Lock 只能由 OSPM 或固件独占。当 Lock 所有权被申请时，它可能被其它设备占用，在此情况下申请环境退出并且等待 Lock 被释放的信号发出。例如 Global Lock 可以用来保护 Embedded Controler 接口。在同一时刻只能有一个 OSPM 或者固件访问 Embedded Controler 接口。有了这把“锁”可以有效地防止设置风扇转速时发生的冲突导致死机等情况发生。因全局锁的获取和释放不是本文论述的重点，因此只作简略说明，不深入说明。

在所列实例中 LockRule 为“NoLock”表示设置转速中不需要加锁。因为得到的以上代码是硬件底层代码，如果在操作系统下调用有必要在两次端口操作之间加入延时以增强稳定性，延时的长短可以是 2-10 毫秒，得到函数 SetFanOn 设置风扇转速的最终方法如下。

在所列实例中 LockRule 为“NoLock”表示设置转速中不需要加锁。因为得到的以上代码是硬件底层代码，如果在操作系统下调用有必要在两次端口操作之间加入延时以增强稳定性，延时的长短可以是 2-10 毫秒，得到函数 SetFanOn 设置风扇转速的最终方法如下。

```
void SetFanOn(unsigned char speed)
{
    outportb(0x3E, 0x92);
    delay(10);
    outportb(0x3F, speed);
}
```

本实施例只详细描述了风扇某一种状态“_ON”方法 C++代码的获取生成过程，其它方法与此方法类似，只需根据本发明采用相同的方式就可以得到全部实现。

当然，本发明还可有其他多种实施例，在不背离本发明精神及其实质的情况下，熟悉本领域的技术人员可根据本发明作出各种相应的改变和变形，但这些相应的改变和变形都应属于本发明权利要求的保护范围。

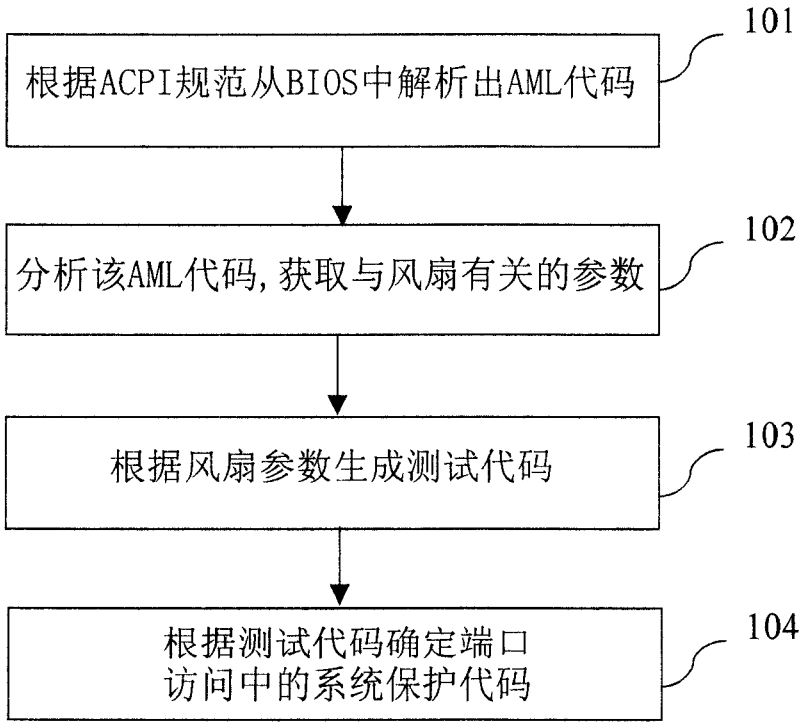

101
根据ACPI规范从BIOS中解析出AML代码
102
分析该AML代码,获取与风扇有关的参数
103
根据风扇参数生成测试代码
104
根据测试代码确定端口
访问中的系统保护代码

图1

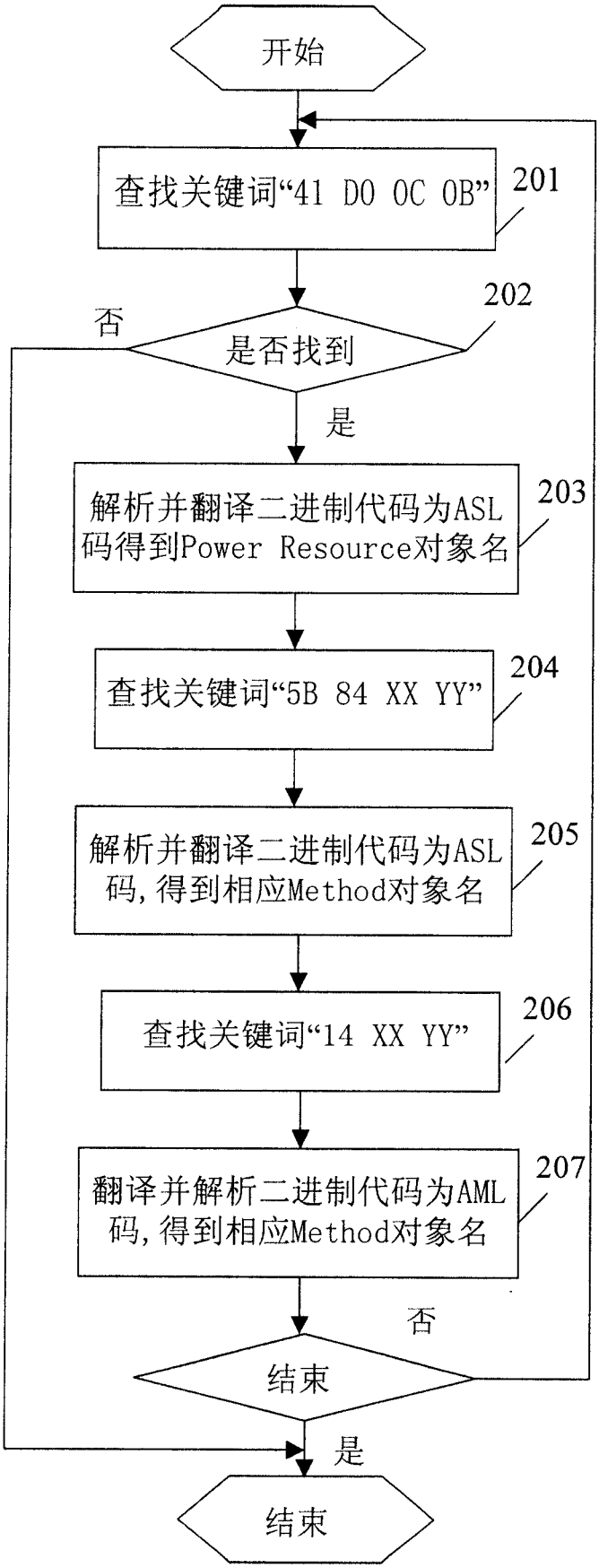

开始
查找关键词“41 D0 0C 0B”
201
202
否
是否找到
是
解析并翻译二进制代码为ASL码得到Power Resource对象名
203
查找关键词“5B 84 XX YY”
204
解析并翻译二进制代码为ASL码，得到相应Method对象名
205
查找关键词“14 XX YY”
206
翻译并解析二进制代码为AML码，得到相应Method对象名
207
否
结束
是
结束

图2