改写 PCI 网卡的 MAC 地址的原理与技术

The Principle and Technique to Write MAC Address of PCI Network Card

李全忠 刘长文 王希超 （山东农业大学 信息学院 泰安 271018）

摘要：以 RTL8139 网卡为例，介绍了改写 MAC 地址的原理，给出了相应的技术细节和基于 Delphi 语言的实现程序，从而较好地解决了在应用程序中直接改写 MAC 地址的技术问题。

关键词：PCI 总线 RTL8139 网卡 AT93C46 芯片 MAC

1 引言

网卡的 MAC 地址是一个 6 字节（48 位）的地址标识，在网络通讯中起到区分节点的作用，有时也称其为节点地址。网卡的 MAC 地址一般情况下是不应改变的，否则会造成网络通讯混乱或失败。然而，当面临下列情况时又要另当别论了：比如遭遇雷击或强磁场干扰改变了网卡的 MAC 地址，再比如某种病毒改写了网卡的 MAC 地址，等等。

显然，为了保证 MAC 与 IP 的绑定在本域的有效性，可行的办法之一就是将正确的 MAC 地址重新写入，这样做的优点在于，省去了重新办理绑定手续的费时费力的琐事。那么，作为一般用户怎样改写网卡的 MAC 地址？下面以较为流行的 RTL8139 网卡为例进行探讨。

2 改写 MAC 地址的基本思路

RTL8139 网卡的 MAC 地址存储在其上的 AT93C46 串行芯片中，它是一个电擦写可编程只读存储器 EEPROM。从系统和技术层面上讲，改写 MAC 地址的总体思路是：（1）获得 RTL8139 网卡的 IO 空间基址；（2）获得 AT93C46 的命令寄存器口地址；（3）改写 AT93C46 存储器中的 MAC 地址单元的内容。

3 改写 MAC 地址的实现过程

3.1 获得 RTL8139 网卡的 IO 空间基址

RTL8139 网卡同其它 PCI 总线设备一样，都有一个配置空间。简言之，配置空间是一个容量为 256 字节具有特定记录格式的地址空间。当电脑启动时，BIOS 便对 PCI 设备的配置空间进行合理配置，包括对一些寄存器设置和存储器映射等。只有经过配置后的配置空间才能进行读写操作。表 1 列出了 RTL8139 网卡配置空间中的关键字段。

表 1 RTL8139 网卡配置空间中的关键字段

偏移量	长度	功能描述
00 – 01H	字	主控芯片制造商标志 VID，只读，RTL8139 的 VID = 10EC
02 – 03H	字	设备标志 DID，只读，RTL8139 的 DID = 8139
10 – 13H	双字	IO 空间基址寄存器，可读写

这里的 VID、DID 字段是用来识别特定 PCI 设备的标志；对于 PCI 网卡而言，偏移 10 – 13H 是网卡的 IO 空间基址寄存器。

对配置空间的读写操作，是通过 PCI 协议规定的两个特定的 32 位 IO 寄存器 CF8H 和 CFCH 实现的。CF8H 称为配置地址寄存器，用于产生配置空间的地址，CFCH 称为配置数据寄存器，用于保存配置空间的读写数据。配置地址寄存器格式如图 1。

31	30 – 24	23 – 16	15 – 11	10 – 8	7 – 2	1 – 0
使能位	保留	总线号	设备号	功能号	寄存器号	00

图 1 配置空间地址寄存器格式

在配置地址寄存器中，当使能位为 1 时表示可以对配置空间进行读写操作；总线号为系统的 256 个总线之一，在 AT 系列机指 PCI 插槽的总线号，一般为 0 – 5；设备号为 PCI 设备的编号；功能号用区分多功能设备，单功能网卡设备为 000b；6 位寄存器号用于寻址该 PCI 设备配置空间 64 个双字寄存器。其中，RTL8139 的

IO 空间基址由其配置空间的偏移 10 – 13H 处的双字寄存器给出。

应当说明的是，PCI 设备的 IO 空间基址最低两位读出时固定为 01b，对地址本身无效，应当屏蔽掉，对于 AT 机 IO 有效地址为 16 位，因此高 16 位也应该屏蔽掉，编程时必须注意。

在 Delphi 语言中若令变量 PCIID 取值为 "813910EC"，即 RTL8139 网卡的 VID 和 DID 标志，则下面的函数返回的是 RTL8139 网卡的 IO 空间基址 Get8139IO(PCIID)。

```
Function TForm1. Get8139IO(PCIID:String):Word;
var
  Bus,Device:Byte;
  O_Add,I_Dat:Integer;
begin
  Result:= $ ffff;
  for Bus:=0 to 5 do
    for Device:=0 to 31 do
      begin
        O_Add:= $ 80000000 + Bus * $ 10000 +
(Device * 8) * $ 100;
        asm    mov    dx, $ cf8
               mov    eax,O_Add
               out    dx,eax
               mov    dx, $ cfc
               in     eax,dx
               mov    I_Dat,eax
        end;
        if IntToHex(I_Dat,8) = PCIID then // 若
RTL8139 网卡存在
          begin
          O_Add:= $ 80000000 + Bus * $ 10000 +
(Device * 8) * $ 100 + $ 10;
          asm    mov    dx, $ cf8
                 mov    eax,O_Add
                 out    dx,eax
                 mov    dx, $ cfc
                 in     eax,dx
                 and    eax, $ fffe // 屏蔽高 16 位
及低 2 位
                 mov    @Result,ax // 返回 IO 空
间基址
          end;
        end;
      end;
end;
```

3.2 获得 AT93C46 存储器的命令寄存器口地址

分析表明，AT93C46 存储器的命令寄存器 93C46CR 是一个 8 位的寄存器，其口地址被映射到 RTL8139 网卡 IO 空间基址的偏移 50H 上。为了以后使用的方便，我们用 CR93C46 表示 93C46CR 的口地址，即 CR93C46 = Get8139IO(PCIID) + $ 50。

3.3 改写 AT93C46 存储器中的 MAC 地址单元

3.3.1 AT93C46 存储器的主要特征

AT93C46 是一个 1K 位的串行 EEPROM 存储器，它通过芯片选择管脚(CS)来启动，并经一个含数据输入(DI)、数据输出(DO)和移动时钟(SK)的 3 线串行接口存取。在读数据时，读指令地址被解码，数据在数据输出管脚上随时钟串行输出。写循环完全可以自定时，并且写之前不需单独的擦除循环，写循环只能在器件与擦除/写使能状态时工作。当写循环被激发，且 CS 置高电平时，DO 管脚输出器件忙/闲工作状态。

在 RTL8139 网卡上，AT93C46 的管脚 ORG 与 VCC 相连，其内部组态为 64 × 16 位，即 16 位存储模式，也即以字为单位进行编址和寻址，寻址范围为 0 – 63。其中网卡的 MAC 地址被存储在它的第 7 – 9 连续的 3 个字中，每个字包含 MAC 地址中的两个字节，且低字节在前高字节在后。这 3 个字就是我们要改写的对象。

掌控 AT93C46 读写操作的是它的命令寄存器 93C46CR。它是一个 8 位寄存器，其功能见表 2。

由表 2 知道，AT93C46 命令寄存器 93C46CR 的低 4 位与 AT93C46 芯片的管脚一一对应，因此可以通过读写 AT93C46 的命令寄存器来读写 AT93C46 存储器，从而达到改写 MAC 地址的目的。

由于 AT93C46 是以字为单位进行存储的，故对其内容的改写必须以字为单位进行，即每次只能写入 1 个字(2 个字节)。因此，完整的 MAC 地址的写入需要执行 3 次独立的写过程。

3.3.2 向 AT93C46 存储器写入字的技术细节

依据 AT93C46 的写规范，向 AT93C46 写入数据时

每次只能写入 1 个字，而这个写入过程要执行 3 个 AT93C46 命令：(1) 写使能 (EWEN)；(2) 写数据 (WRITE)；(3) 写禁止 (EWDS)。而每执行完一个命令后，还需要向命令寄存器 93C46CR 写入一个 0。相应的命令描述见表 3。

表 2　AT93C46 的命令寄存器功能描述

位	符号	功能描述
7－6	EEM1－0	=10b，表示 93C46 编程模式
5－4	－ － －	不用
3	EECS	对应 93C46 的 CS(Chip Select) 脚
2	EESK	应 93C46 的 SK(Serial Data Clock) 脚
1	EEDI	对应 93C46 的 DI(Serial Data Input) 脚
0	EEDO	对应 93C46 的 DO(Serial Data Output) 脚

表 3　AT93C46 的部分相关命令描述

命令	开始位	操作码	地址	数据
写使能 (EWEN)	1	00	11XXXX	
写数据 (WRITE)	1	01	A5－A0	D15－D0
写禁止 (EWDS)	1	00	00XXXX	

表 3 中的 $A_5－A_0$ 表示某一字 (16 位) 内容的地址单元，寻址范围为 0－63，以字节的低 6 位表示；$D_{15}－D_0$ 表示要写入的字 (16 位) 内容。

根据上述命令描述和相应的时序电路 (因篇幅所限，时序电路图略) 可知，相应命令代码的二进制数据分别为：

写使能 =1 00111100；

写数据 =1 01 $A_5－A_0$　$D_{15}－D_8$　$D_7－D_0$；

写禁止 =1 00000000。

下面用 Delphi 编写的过程 Write_MAC(Addr:Byte; MAC_W:Word)，实现了向指定地址 Addr 写入 1 个字内容为 MAC_W 的操作。其中 Out_93C46(Value:Byte) 是向 AT93C46 移位输出 8 位数据 Value 的过程，Out_93C46_1 是向 AT93C46 输出 1 位数据 1 的过程，Out_93C46CR_0 是向命令寄存器 93C46CR 写 0 的过程。Delay 是延迟过程，用以确保 AT93C46 每一步写操作的成功执行。相关技术细节在后附的代码中均有注释。另外，需要说明的一点是，由于适时增加了延迟处理，就略去了字内容写入后 AT93C46 是否处于忙状态的测试。

具体地说，若在 Delphi 应用程序中执行了如下代码：

```
Procedure TForm1. Button1Click(Sender: TObject);
var PCIID:string;
begin
  PCIID:='813910EC';
  CR93C46:=Get8139IO(PCIID)+$50;
  Write_MAC(07,$0012);
  Write_MAC(08,$3456);
  Write_MAC(09,$78AB);
end;
```

则本机 RTL8139 网卡的 MAC 地址变为 00－12－34－56－78－AB 了。

下面的代码是 Write_MAC 过程以及它所调用的子过程。需要说明的是，在 Delphi 环境下运行这些代码时，须在主程序的首部声明这些相应的函数和过程，并且把变量 CR93C46 设置为模块变量或全局变量。

```
Procedure TForm1. Write_MAC(Addr:Byte; MAC_W:Word);
begin
  //写使能 =1 00111100
  Out_93C46_1;
  Delay;
  Out_93C46($3C);
  Delay;
  Out_93C46CR_0;
  Delay;
  //写数据 =1 01 A5－A0  D15－D8  D7－D0
  Out_93C46_1;
  Delay;
  Out_93C46($40+Addr);
  Delay;
  Out_93C46(Lo(MAC_W));
  Delay;
  Out_93C46(Hi(MAC_W));
  Delay;
  Out_93C46CR_0;
  Delay;
  //写禁止 =1 00000000
  Out_93C46_1;
```

```
  Delay;
  Out_93C46(0);
  Delay;
  Out_93C46CR_0;
  Delay;
end;
Procedure TForm1. Out_93C46(Value:Byte);
begin
  asm
  // 7 6 5 4 3 2 1 0
  //eem1 eem0 nc nc eecs eesk eedi eedo(93C46 命令寄存器)
  // cs sk di do(AT93C46 管脚)
      mov     cx,8
      mov     dx,CR93C46
      mov     ah, Value
  @@L0: shl ah,1
      jc @@L1
  //out 0b
      mov al,88h // 1000 1000b:cs=1,sk=0,di=0
      out dx,al
      mov al,8ch // 1000 1100b:cs=1,sk=1,di=0
      out dx,al
      jmp @@L2
  //out 1b
  @@L1: mov al,8ah // 1000 1010b:cs=1,sk=0,di=1
        out dx,al
        mov al,8eh // 1000 1110b:cs=1,sk=1,di=1
        out dx,al
  @@L2: loop @@L0
  end;
end;
Procedure TForm1. Out_93C46_1;
begin
  asm
  // 7 6 5 4 3 2 1 0
  //eem1 eem0 nc nc eecs eesk eedi eedo(93C46 命令寄存器)
  //          cs sk di do(AT93C46 管脚)
        mov dx,CR93C46
        mov al,8ah // 1000 1010b:cs=1,sk=0,di=1
        out dx,al
        mov al,8eh // 1000 1110b:cs=1,sk=1,di=1
        out dx,al
  end;
end;
```

4　结束语

显见，了解了 MAC 地址的改写原理和具体的实现技术，用户便可在应用程序中及时更正被改写的 MAC 地址，使相关绑定在本域继续有效，达到省事省时的目的。本文所给代码在 Win9x 下运行通过。

参考文献

1　李全忠、岳训、费玉奎，直接读取网卡节点地址的原理与方法[J]，计算机工程，2003. 14. 181－182.

2　李全忠、王希超，直接读取 PCI 网卡的 MAC 地址的原理与方法[J]，计算机工程，2005. 18. 213－215.

3　余永全，Flash 单片机原理及应用[M]，北京：电子工业出版社，1999.