MAC Address Cloning

kingpin@l0pht.com
12.25.98

What's the point?

At one time or another, you may find it necessary to have two machines with the same physical network address. This could be for a number of reasons: testing of your internal network or software, attempting network snooping or spoofing, bypassing computer software security that relies on this address or just plain curiosity. Whatever the reason, what follows is how it's done.

Each computer attached to an Ethernet is assigned a unique 48-bit (6-byte) integer, known as an Ethernet or MAC (Media Access Control) address. Ethernet NIC (Network Interface Card) hardware manufacturers purchase blocks of Ethernet addresses from the IEEE (Institute for Electrical and Electronic Engineers) and assign them in sequence to their cards.[1] The IEEE global address ranges are assigned and registered by IEEE to individual companies requesting them. Every company is responsible for ensuring that every manufactured unit gets a unique address within its assigned range of addresses.[2] Thus, no two cards will purposely have the same address. The first 3-bytes of the address are used to identify the manufacturer of the NIC. The last 3-bytes are the unique serial number. A list of vendor codes can be found at http://MAP-NE.dragonfire.net/Ethernet/vendor.html.

In almost any available NIC, the MAC address is stored in semiconductor-based memory containing one or more of the following:

- MAC Address
- I/O Base Address
- Interrupt
- Interface Type (10BaseT, AUI, 10Base2)
- Checksum
- Other configuration/initialization information

The earlier cards used a now obsolete 82S123 or 74S288 one-time programmable 256-bit device. These devices were large (16-pin through-hole), costly and difficult to program.

Almost all of today's NICs use a Serial EEPROM[3] (Electrically Erasable Programmable Read-Only Memory) to store this information. Often socketed, but usually soldered to the board, these devices use a three-wire interface, known as Microwire, to transfer its contents. The EEPROMs are programmed during card manufacture and placed onto the board. In some cases, the Serial EEPROM is programmed directly from the Ethernet controller IC, so that the MAC address and other configuration information can be entered via PC software. The beauty of these devices is that they can be erased and written up to 1 million times without error. Due to the limited amount of memory required, these devices often have only 256-bit (93C06) or 1024-bit (93C46) capacity. Our work concentrates on modifying the contents of this Serial EEPROM to change the necessary MAC address and Checksum data.

The heart of a NIC is an "Ethernet Interface" or "Ethernet Controller" IC. This device handles the necessary interfacing between the Ethernet and PC bus. This chip really "runs the show", as the other components on the NIC are usually just interface/glue logic. There are many different manufacturers of Ethernet controllers. Our cards used a National Semiconductor DP83905-EB AT/LANTIC[4], which interfaces to an ISA bus. Other devices include Realtek RTL8129 and RTL8029 Fast Ethernet Controller[5] designed for interface to a PCI bus, the Standard

[1] Internetworking with TCP/IP, Vol. 1, Douglas Comer, pg. 25, ISBN 0-13-468505-9
[2] SMSC LAN-9000 FAQ, http://www.smsc.com/main/appnotes/tn76.html
[3] Basic Serial EEPROM Operation, http://www.microchip.com/Download/Appnote/Category/EEPROMS/00536.pdf
[4] DP83905EB-AT AT/LANTIC Hardware User's Guide, http://www.national.com/an/AN/AN-897.pdf
[5] RTL8129 Preliminary Data Sheet, http://www.realtek.com.tw/cn/NEW/doc/RTL8129-new.htm

Microsystems LAN9000-family[6] and the UMC UM9003. In most cases, this Ethernet controller interfaces directly with a Serial EEPROM to store its re-programmable data.

This paper is based on PC-based NICs using the AT/LANTIC device, which emulates the extremely common Novell NE2000 Plus card. Our initial research was performed before we came upon the AT/LANTIC data sheet, but the results match closely and helped us verify the contents of the EEPROM.

How do I find out what my MAC address is?

Frequently, the MAC address of the NIC is printed on a label and stuck to the card itself. If the card you want to clone isn't accessible, you'll probably want a non-intrusive method of obtaining the address. Depending on the particular flavor of your operating system, the method of finding your MAC address varies[7].

What's on the Serial EEPROM?

Our initial experiments consisted of simply reading data from the Serial EEPROM on a number of different NICs. By doing so, we tried to see the likeness and difference between them, and verified the storage of the MAC address. We could easily have determined this by looking at the data sheet for the Ethernet controller IC, but keep in mind that we didn't have one at this stage of the experiment. Table 1 shows the comparison.

Table 1 – NIC Serial EEPROM comparison

Manufacturer	Model	EEPROM	MAC Address	Data
National Semiconductor	NSC ?	93LC06	08:00:17:03:C0:E5	**0008 0317 E5C0** 0000 0500 010D 01DA **5757 4242** 0000 0000 0000 0000 0000 0020 0020
Ansel Communications	N2000 Plus 3	93C46	00:40:90:80:07:7E	**4000 8090 7E07** FFFF FFFF FFFF FFFF **5757 4242** FFFF FFFF FFFF FFFF FFFF 0100 FF20
Microdyne	NE2000 Plus 3	93C06	00:80:29:E7:C2:9C	N/A
Linksys	Ether16	93C46	00:40:05:44:17:A7	**4000 4405 A717** 0108 020A 5464 00D8 0000 0000 0000 0000 0000 0000 0000 0000 0000
Genius	GE2000 II	93C46	00:40:33:2A:82:82	**4000 2A33 8283** 5805 0000 0000 0000 **5757 4242** 0000 0000 0000 0000 0000 2100 0020
Winbond	HT-2003CT	93C46	48:54:33:01:48:24	**5448 0133 2448** 0000 5448 0133 2448 **5757 4242** 0000 0000 0000 0000 0000 4040 0020

Looking at the table, it's easy to determine where and how the MAC address is stored. The nibbles are reversed, but the MAC address consists of bytes 0-5. A common trait to most of the data is the 5757 4242 string at byte 15.

For the NICs using a 93C46 EEPROM, the 32-bytes of data were either repeated four times (to fill the 128-bytes of capacity) or just filled with 0x00. The amount of data stored in the EEPROM is dependent on the Ethernet controller IC. Many new controllers are designed to interface to a 93C46 only. The migration to the 93C46 was due to the availability and production status of the 93C06, which has been phased out.

If you haven't noticed, the data is missing from the Microdyne NE2000 NIC. This was the first card I attempted to read and, due to the excitement, I wrote over the existing data right after I read it. As you laugh, remember that the Serial EEPROM devices are re-programmable, so take caution as you handle the card. By looking at the data from the other cards and knowing the MAC address, we can assume that the first 3 words of the device will be: 8000 E729 9CC2.

That's easy. What else?

So we know that the MAC address is stored in the EEPROM. We also know that there is a 5757 4242 string common to some of the devices. There is some other data that varies between the devices. Now what do we do?

Behold! This is where the AT/LANTIC data sheet comes into play. Section 1.3 "EEPROM Programming" of the AT/LANTIC data sheet and Table 2 describes the memory map of the Serial EEPROM.

[6] LAN9000 Product Datasheets, http://www.smsc.com/main/datasheet.html
[7] How to find your MAC address, http://www-dcg.fnal.gov/DCG-Docs/mac/

Table 2 – AT/LANTIC Serial EEPROM Memory Map

Addr	Bits 15-8	Bits 7-0
00	MAC Address 1	MAC Address 0
01	MAC Address 3	MAC Address 2
02	MAC Address 5	MAC Address 4
03	Checksum	05 (8013 type)
04	00 (Not Used)	00 (Not Used)
05	00 (Not Used)	00 (Not Used)
06	00 (Not Used)	00 (Not Used)
07	57 (ASCII “W”)	57 (ASCII “W”)
08	42 (ASCII “B”)	42 (ASCII “B”)
09	00 (Not Used)	00 (Not Used)
0A	00 (Not Used)	00 (Not Used)
0B	00 (Not Used)	00 (Not Used)
0C	00 (Not Used)	00 (Not Used)
0D	00 (Not Used)	00 (Not Used)
0E	Config B	Config A
0F	73 (For future use)	Config C

On most NICs, the Serial EEPROM can be used to implement a jumper-less solution. The EEPROM will store all necessary configuration and initialization information, including I/O address and interrupts, replacing the need to set physical jumpers. A NIC with a jumper-less configuration requires the use of the Config A, B and C bytes. Section 1.1.9 “Configuration” of the AT/LANTIC data sheet explains this in detail. Although these bits don’t need to be modified to clone the MAC address, for inquiring minds, Table 3 shows what the bit fields represent.

Table 3 – AT/LANTIC Configuration bits

Config A:

Bit	Use
0	I/O Address
1	I/O Address
2	I/O Address
3	Interrupt
4	Interrupt
5	Interrupt (if 8 selected)
6	Fast read
7	NE2000/Shared Memory

Config B:

Bit	Use
0	AUI/Coax/TPI
1	AUI/Coax/TPI
2	Good Link Test Disable
3	IO16 Bug Fix Enable
4	IO CHRDY Bug Fix Enable
5	-
6	Boot PROM Write Enable
7	-

Config C:

Bit	Use
0	Boot PROM Address & Size
1	Boot PROM Address & Size
2	Boot PROM Address & Size
3	Boot PROM Address & Size
4	RAM Size 8k or 32k
5	4 or 8 Interrupts
6	Core CLK = 20MHz or BSCLK
7	Allow access to configure registers

Let’s compare.

Using the fifth entry from Table 1, the Genius GE2000 II, let’s compare our obtained data with the expected data from Table 2. By doing this, we should be able to match up byte for byte the information in the Serial EEPROM, and determine what needs to be programmed to change the MAC address. By now, it should be obvious that the only thing that needs to be changed in the Serial EEPROM is the MAC Address and Checksum, but we’ll do this exercise just for fun.

Table 4 – Breakdown of Genius GE2000 II data by byte

Location	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Data	40	00	2A	33	82	83	58	05	00	00	00	00	00	00	57	57

Location	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Data	42	42	00	00	00	00	00	00	00	00	00	00	21	00	00	20

As we already know, the first 6 bytes (location 0-5) are the MAC address.

The next byte (location 6) is the Checksum. The Checksum is calculated so that the least significant byte of the sum of the first 8 bytes in the EEPROM is 0xFF. In this case, it's 0x58. Let's verify:

0x40 + 0x00 + 0x2A + 0x33 + 0x82 + 0x83 + 0x05 + Checksum (0x58) & 0xFF = 0xFF

All the bytes match up through location 27. Three of the remaining bytes consist of the configuration bytes, which will vary from card to card.

Config A (Location 29) = 0x00
Config B (Location 28) = 0x21
Config C (Location 31) = 0x20

Location 30 is specified to have a 0x73 written to it. All the cards we've looked at had either 0x00 or 0xFF. This byte is said to be used for future expansion of features on the AT/LANTIC chip.

Serial EEPROM Programming

To change the MAC address of a NIC card, you'll need to reprogram the Serial EEPROM to reflect a new MAC address and Checksum byte, at minimum. Most any off-the-shelf device programmer can handle Serial EEPROMs and there are a handful of simple Parallel Port and RS232 programmers available, as well. Simply find the data sheet for the particular Ethernet controller IC on your NIC, and work from there.

Microchip produces a "Serial EEPROM Designers Kit", which I highly recommend for purposes of this experiment. The kit is a serial-port based programmer, which includes a "Windows and DOS development system that programs all of Microchip's Serial EEPROM memories, including Microwire, I2C and SPI protocols."[8] The necessary software and manuals are offered free-of-charge on the web[9]. Contact Microchip directly for product pricing and availability.

If you want a more robust device programmer, for use with other projects, there are quite a available on the market. Most of these are focused on microprocessor and larger memory device programming, but they all offer support for Serial EEPROMs. Table 5 consists of reliable and (sometimes) low-cost programmers, based on personal experience.

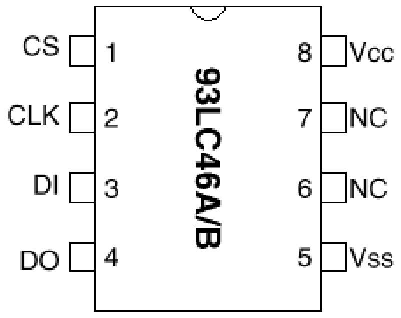
Table 5 – Commercial-grade device programmer resources

Manufacturer	Web Page	Model Number
BP Microsystems	http://www.bpmicro.com	Any, BP-1148, BP-1200
Data I/O	http://www.dataio.com	Any, ChipWriter Portable
M2L Electronics	http://www.m2l.com	EZ-EP

If you feel the need to "roll your own" programmer, keep reading. Interface to a Serial EEPROM requires a minimal amount of connections. Figure 1 shows the pinout for an 8-pin DIP Microchip 93C46 device[10]. The data sheet digs a little deeper.

[8] Serial EEPROM Designers Kit Overview, http://www.microchip.com/10/Tools/memory/serial/index.htm
[9] http://www.microchip.com/Download/Tools/Memory/Serial/evlsetup.zip
[10] 93LC46 64 x 16 Serial EEPROM Data Sheet, http://www.microchip.com/Download/Lit/Memory/Micro/21173e.pdf

Figure 1 – Typical pinout for Serial EEPROM (DIP package)



The device consists of three input lines: Chip Select (CS), Clock (CLK) and Data In (DI), and one output line: Data Out (DO). All necessary timing diagrams are described in detail in the device data sheet and application notes.

Table 6 – Homebrew device programmer resources

Title	Web Page
Microchip Serial EEPROM Application Notes	http://www.microchip.com/10/Appnote/Category/EEPROMS/index.htm
Parallel Port - Programming Serial EEPROM	http://www.phanderson.com/printer/eeprom/eeprom.html

All of our NICs, with the exception of one, had the Serial EEPROM soldered to the board. I would suggest removing the device and replacing it with a DIP socket of some type, to allow easy removal later on. De-soldering the chip from the board can be easy with the right tools, and not so easy with the wrong ones. Be careful not to crack pins or de-laminate the PCB by excessive heating. I have not had luck programming these devices "in-circuit", so removal is necessary.

Do any non-hardware solutions exist?

There are a few possibilities for modifying the MAC address without mucking with the hardware. Using SunOS, for example, the settings can be changed via the ifconfig (1M) command. On SPARCs, you can set it in NVRAM using the prom-monitor.

As discussed earlier, there is other information being reprogrammed into the Serial EEPROM directly from the Ethernet controller IC (via some host-based software) for use in a jumperless configuration. Because of this, there is no reason that the MAC address can't be reprogrammed from the host-based software, as well. For you software folks, it might be an interesting task to reverse engineer the setup/configuration software that comes with your jumperless NIC and hack it to reprogram the MAC address (if it doesn't, already).

I want more information!

Oh.

-kp