OM-O2S / OM-O2SP

Onion Omega2S IoT compute modules

Device MAC Addresses & Unique IDs

R01

Application Note

Abstract

This document describes how to programmatically generate a unique identifier for an Omega2 IoT computer based on the device's MAC address. Along the way, MAC address allocation during production, how to read the device's MAC addresses, and mapping of MAC addresses to networking interfaces is discussed.

Device MAC Addresses

The Omega2, being an IoT Computer, is a connectivity device. Here we'll discuss the Omega2's MAC addresses, how they're allocated, how to read them, and how to programmatically generate a unique ID based on them.

> A MAC (Media Access Control) address is a unique, 6-byte identifier assigned to network interface controllers. Devices with multiple network interfaces must have a unique MAC address for each interface.

MAC Address Allocation

Every Omega2 device is allocated **three** sequential MAC address during production, one for each of the available network interfaces:

- `ra0` – the WiFi Access Point interface – this matches the MAC address on the sticker on the shielding
- `eth0` – ethernet port
- `apcli0` – the WiFi client interface

The `ra0` MAC address matches the MAC address that is printed on the device's shielding. The other two MAC addresses are sequential:

- The `eth0` MAC address is the `ra0` MAC address + `0x01`
- The `apcli0` MAC address is the `ra0` MAC address + `0x02`

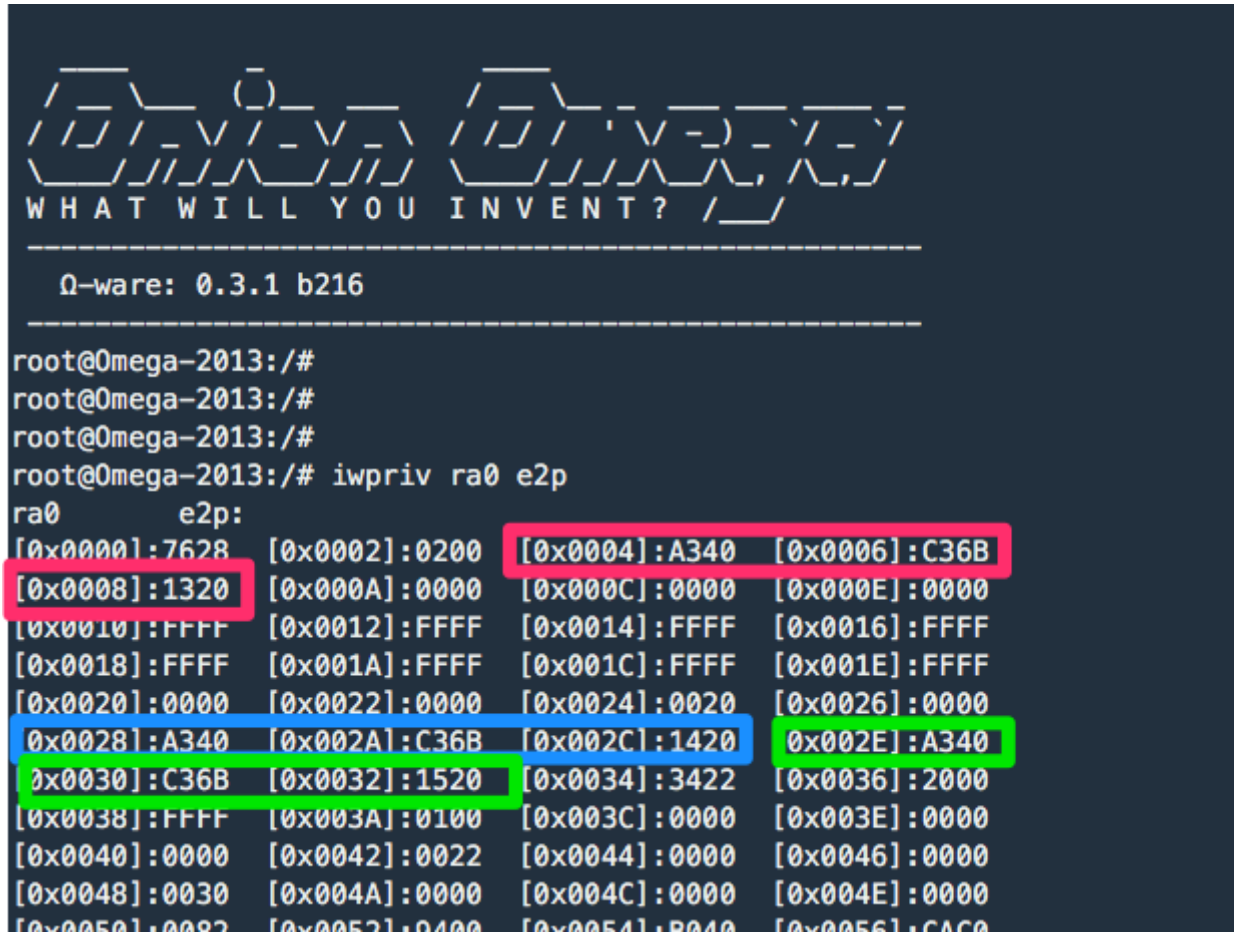
Reading the MAC Address

The `iwpriv` command can be used to access the wireless driver parameters, among these parameters are the MAC addresses.

Connect to the Omega's command line and run the following command:

```
iwpriv ra0 e2p
```

It will output quite a bit of data:



We're interested in:

- The 6 bytes starting at `0x0004` - the `ra0` MAC address
- The 6 bytes starting at `0x0028` - the `eth0` MAC address
- The 6 bytes starting at `0x002e` - the `apcli0` MAC address

Take a look at how these address map to the network interfaces by running the `ifconfig` command:

```
root@Omega-2013:/# ^C
root@Omega-2013:/# ^C
root@Omega-2013:/# ifconfig
apcli0    Link encap:Ethernet  HWaddr 40:A3:6B:C3:20:15
          inet6 addr: fe80::42a3:6bff:fec3:2015/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

br-wlan   Link encap:Ethernet  HWaddr 40:A3:6B:C3:20:13
          inet addr:192.168.3.1  Bcast:192.168.3.255  Mask:255.255.255.0
          inet6 addr: fe80::42a3:6bff:fec3:2013/64 Scope:Link
          inet6 addr: fd1d:48c4:7633::1/60 Scope:Global
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:6063 (5.9 KiB)

eth0      Link encap:Ethernet  HWaddr 40:A3:6B:C3:20:14
          inet6 addr: fe80::42a3:6bff:fec3:2014/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:157 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:50562 (49.3 KiB)
          Interrupt:5

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:164 errors:0 dropped:0 overruns:0 frame:0
          TX packets:164 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:10660 (10.4 KiB)  TX bytes:10660 (10.4 KiB)

ra0       Link encap:Ethernet  HWaddr 40:A3:6B:C3:20:13
          inet6 addr: fe80::42a3:6bff:fec3:2013/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:6
```

Programmatically Generate a Unique ID based on the Device's MAC Address

When working with IoT devices, it's always useful when an ID that's unique to each device can be generated with the same code. And what better to use than the device's own MAC address.

To illustrate the process, consider a shell script named `macId.sh`. The shell script will create and output a unique ID based on the Omega's `ra0` MAC address:

```
#!/bin/sh

## Generate UID based on device's MAC addr for ra0 intf

generateMacUid () {
        # grab line 2 of iwpriv output
        line1=$(iwpriv ra0 e2p | sed -n '2p')
        # isolate bytes at addresses 0x0004 and 0x0006, and perform byte swap
        bytes5432=$(echo $line1 |  awk '{print $3":"$4}' | awk -F ":" '{print subst

        # grab line 3 of iwpriv output
        line2=$(iwpriv ra0 e2p | sed -n '3p')
        # isolate bytes at address 0x0008 and perform byte swap
        bytes10=$(echo $line2 | awk '{print $1}' | awk -F ":" '{print substr($2,3)

        macId=$(echo ${bytes5432}${bytes10})
        echo $macId
}

uid=$(generateMacUid)
echo "Unique ID based on MACD for this device is: $uid"
```

When run on the device:

```
root@Omega-2013:~# sh macId.sh
Unique ID for this device is: 40A36BC32013
```

Feel free to translate this code to other languages and use it in your own applications.