# *MBARI Medusa*
# *Nvstore Initialization Procedure*

## *Introduction*

The Medusa card contains a significant amount of information stored in a eeprom on the board. When a board is fresh from the factory, this eeprom needs to be initialized with board level information such as serial number and ECO level as well as the important IP address parameters for the resident management agent. A software module has been written, referred to as the 'nvstore', to allow easy and robust access to information stored in non-volatile memory. Each information item is protected by a 16-bit CRC.

## *Nvstore system Initialization Procedure*

When powering up a Medusa for the first time, you should see a display like this on the serial port:

```
MBARI Medusa v 2.40 (Rowley gcc 3.4.4) Aug 16 2005 14:39:57

-> ! no CPU MAC address is stored -- use setmac to set one
!! no valid cpu IPv4 address is stored
!! no cpu default gateway is stored
!! no cpu IP netmask is stored
!! no board serial info is stored
!! no board ECO level is stored
```

The nvstore system has it's own initialization command to allow for flexibility in the number of entries. The command to issue at the Medusa command line is nvinit:

```
-> help nvinit
nvinit: nvinit - [numItems] initialize nv store
```

As noted, most commands have a small, but substantive line of help available by just typing 'help' and then the command. Typing 'help' by itself lists all of the available commands. I'll go ahead and initialize the nvstore system to hold 16 items. These items can be up to 60 bytes in length and are identified by an ASCII string. Most of this is done by the modules themselves, and it is unlikely that the user would have to interface with the nvstore system at this level. Let's initialize the nvstore system for sixteen items:

```
-> nvinit 16
AckPoll cnt 3
```

Please ignore the “AckPoll cnt 3” as this is used in debugging, and will be removed.

**The next step is important, but simple. Reset the board.** This can be accomplished by removing power or carefully depressing the reset button in the middle of the board.

## *Configuring a Medusa Board*

After reset, The board will come up and display the same warning messages as before, but the nvstore

system is ready to store values for these items:

```
MBARI Medusa v 2.40 (Rowley gcc 3.4.4) Aug 16 2005 14:39:57

-> ! no CPU MAC address is stored -- use setmac to set one
!! no valid cpu IPv4 address is stored
!! no cpu default gateway is stored
!! no cpu IP netmask is stored
!! no board serial info is stored
!! no board ECO level is stored
```

### *Setting the MAC Address*

Setting the MAC address is usually the first order of business. MBARI has a registered Organization Unique Identifier or OUI of 00:0C:6A. These constitute the first three bytes (in 'wire' or network byte order) of the MAC address. Medusa boards follow this template – 00:0C:6A:01:<board rev>:<board number>, where <board rev> is the Medusa board revision (2 currently), and <board number> is the number given to the board for inventory purposes. The board I am using is serial number 13 (or 0x0d hexidecimal). The last three number should be entered in hexidecimal base. Here is an example:

```
-> help setmac
setmac: setmac [m2] [m1] [m0] - sets CPU MAC address to 00:0c:6a:m2:m1:m0

-> setmac 01 02 0d
AckPoll cnt 3
AckPoll cnt 3
AckPoll cnt 3

-> setmac
setmac: cpu MAC address is 00:0c:6a:01:02:0d
```

Note that the 'setmac' command, invoked by itself, shows the MAC address programmed into the Medusa.

### *Setting up IP Addresses for the TCP/IP Stack*

From here on in, things are very similar. You will be using the ipset, gwset, and maskset commands to set the CPU IP v4 Address, CPU Default Gateway, and CPU Netmask. Note that all IP addresses are expressed as 'dotted quads' or four decimal numbers concatenated together, separated by periods. Below is an example of setting these three TCP/IP stack parameters.

```
-> help ipset
ipset: ipset [IP Addr] - sets Medusa CPU IP v4 address

-> ipset 192.168.0.13
ipset: Medusa CPU IP address is 192.168.0.13
AckPoll cnt 3
AckPoll cnt 3

-> help gwset
gwset: gwset [IP Addr] - sets default gateway

-> gwset 192.168.0.1
```

```
gwset: default gateway is 192.168.0.1
AckPoll cnt 3
AckPoll cnt 3

-> help maskset
maskset: maskset [IP netmask] - sets IP v4 netmask

-> maskset 255.255.255.0
maskset: netmask is 255.255.255.0
AckPoll cnt 3
AckPoll cnt 3
```

Note again, these numbers are in decimal.

### *Board serial number and ECO level setup*

The last non-volatile parameters to be setup currently (not including trunking/failover – this will be covered in a separate document), are the board serial number and ECO level. This allows us to more easily track each Medusa board and what modifications if any have been done to it. These commands are very simple:

```
-> help serial
serial: serial [number] - stores board serial number

-> serial 2.13
AckPoll cnt 3
AckPoll cnt 3
serial: board serial number is [2.13]

-> help ecolevel
ecolevel: ecolevel [level] - stores board ECO level

-> ecolevel A
AckPoll cnt 3
AckPoll cnt 3
ecolevel: board ECO level is [A]
```

Both the serial number and ECO level are stored as strings by the Medusa, so they can be free-form. So far, we have been serializing (assigning a serial number) each board in the format <board rev>.<number>, where <board rev> is the board revision, currently 2, and <number> is the board number, which is assigned to the board by the person or persons who are responsible for keeping the boards in inventory. Recently, this has been Dick Littlefield.

### *Ready for Operation*

If we now reset or power-cycle the Medusa board, all critical non-volatile information has been entered, so no warning messages are generated, and we can see the information that we entered has been retrieved from non-volatile storage by issuing the 'meta' command which provides board metadata on demand:

```
-> [Pushed reset button]
MBARI Medusa v 2.40 (Rowley gcc 3.4.4) Aug 16 2005 14:39:57
```

```
-> meta
CPU MAC address: 00:0c:6a:01:02:0d
CPU IP address: 192.168.0.13
CPU IP netmask: 255.255.255.0
CPU IP gateway: 192.168.0.1
SideARM IP address: 0.0.0.0
Medusa serial number: 2.13
Medusa ECO level: A
Firmware build: v2.40 [Aug 16 2005 14:39:57]
Toolchain: Rowley gcc 3.4.4
```

Of course, we note that the SideARM IP address, used for the selective wakeup service provided by the Medusa at UDP port 6789, has not been set and is initialized to 0.0.0.0 as agreed to in earlier design discussions. To set this address, use the 'saipset' command, as illustrated in this example:

```
-> help saipset
saipset: saipset [IP Addr] - sets SideARM IP v4 address

-> saipset 192.168.0.200
saipset: SideARM IP address is 192.168.0.200

-> meta
CPU MAC address: 00:0c:6a:01:02:0d
CPU IP address: 192.168.0.13
CPU IP netmask: 255.255.255.0
CPU IP gateway: 192.168.0.1
SideARM IP address: 192.168.0.200
Medusa serial number: 2.13
Medusa ECO level: A
Firmware build: v2.40 [Aug 16 2005 14:39:57]
Toolchain: Rowley gcc 3.4.4
```

### *Erasing/Resetting the contents of the nvstore*

If you should ever encounter a need to erase the contents of the nvstore, analogous to reformatting a flash disk, There is an easy procedure to allow this to happen. This involves writing a zero out to location zero (technically offset 0x0) in the EEPROM on Medusa. This is done like this example:

```
-> help eewrite
eewrite: eewrite [offset] [value] - write a single byte

-> eewrite 0 0
eewrite: writing [00] to offset[0000]

-> eeread 0
eeread: read [00] from offset[0000]
```

**At this point reset the board.** It will come up with the standard messages asking you to setup the non-volatile parameters. Now have the equivalent of a board that has come straight from manufacturing and can repeat the steps to initialize the nvstore system described earlier in this document. This can be used to increase the number of original entries specified for non-volatile storage.

```
-> nvinit 32
AckPoll cnt 3
```

Again, **reset the board, as specified following the nvinit command** earlier in this document. You can find out how many entries, the nvstore system is currently configured for by using the nvshow command, as in this example:

```
-> nvshow
nvshow: sig[cf] ver[01] firstItemPg[1] numItem[32]
nvshow: firstDataPg[33] numData[32] configCrc16[ff42]
```

A lot of this information is not useful to anyone but the firmware developer, except for the “numItem” entry above, which shows 32, which is what we have just re-initialized the nvstore system for above.