*A Technote Series on Open Firmware*

# T E C H N O T E: Fundamentals of Open Firmware, Part II: The Device Tree

By Wayne Flansburg
W.FLANSBURG@applelink.apple.com
Apple Developer Technical Support (DTS)

This Technote, the second in a series, describes the Open Firmware device tree. It briefly explains how the device tree is built and then describes some of its contents.

This Technote is targeted at the expansion device designer and the driver writer for that device. The reader should have an understanding of Open Firmware as described by the IEEE 1275-1994 Specification, the PCI Local Bus Specification 2.0, the PCI Bus Binding Specification 1.5, *Designing PCI Cards and Drivers for Power Macintosh Computers*, and *Technote 1044 - PCI Expansion ROM Contents for Mac OS 8, Part III in the Open Firmware Technote Series*.

# Defining the Device Tree

The device tree is an integral part of the Open Firmware 1275-1994 Specification. The Specification defines the device tree as a hierarchical data structure that describes the system hardware and user configuration choices. It also contains hardware drivers and support routines for use by these drivers.

This Technote describes the device tree for the Power Macintosh 9500, which has two AR (Apple RISC) to PCI (Peripheral Component Interconnect) bridge chips called Bandit. Your device tree will look somewhat different than the one described here due to expansion device differences between various machines.

Technote 1061, the first in the series, describes how to connect a host machine to a PCI machine and how to generally work with the user interface for those readers who may need an introduction to the interface.

## *Viewing the Device Tree*

To view the entire device tree, you make the root node the active package. To display the names of the active packages children, enter dev / ls <CR> at the Open Firmware user interface. Hereafter, I will just use interface to describe the Open Firmware user interface. <CR> means type carriage return.

```
Open Firmware, 1.0.5
To continue booting the MacOS type:
BYE<return>
To continue booting from the default boot device type:
BOOT<return>
ok
0 > dev / ls
FF828F80: /PowerPC,604@0
FF829230:   /l2-cache@0,0
FF8299F0: /chosen@0
FF829B20: /memory@0
FF829C68: /openprom@0
FF829D28: /AAPL,ROM@FFC00000
FF829F40: /options@0
FF82A618: /aliases@0
FF82A858: /packages@0
FF82A8E0:   /deblocker@0,0
FF82B0E0:   /disk-label@0,0
FF82B620:   /obp-tftp@0,0
FF82DA60:   /mac-files@0,0
FF82E258:   /mac-parts@0,0
FF82E9B8:   /aix-boot@0,0
FF82EE30:   /fat-files@0,0
FF830400:   /iso-9660-files@0,0
FF830D48:   /xcoff-loader@0,0
FF831708:   /terminal-emulator@0,0
FF8317A0: /bandit@F2000000
```

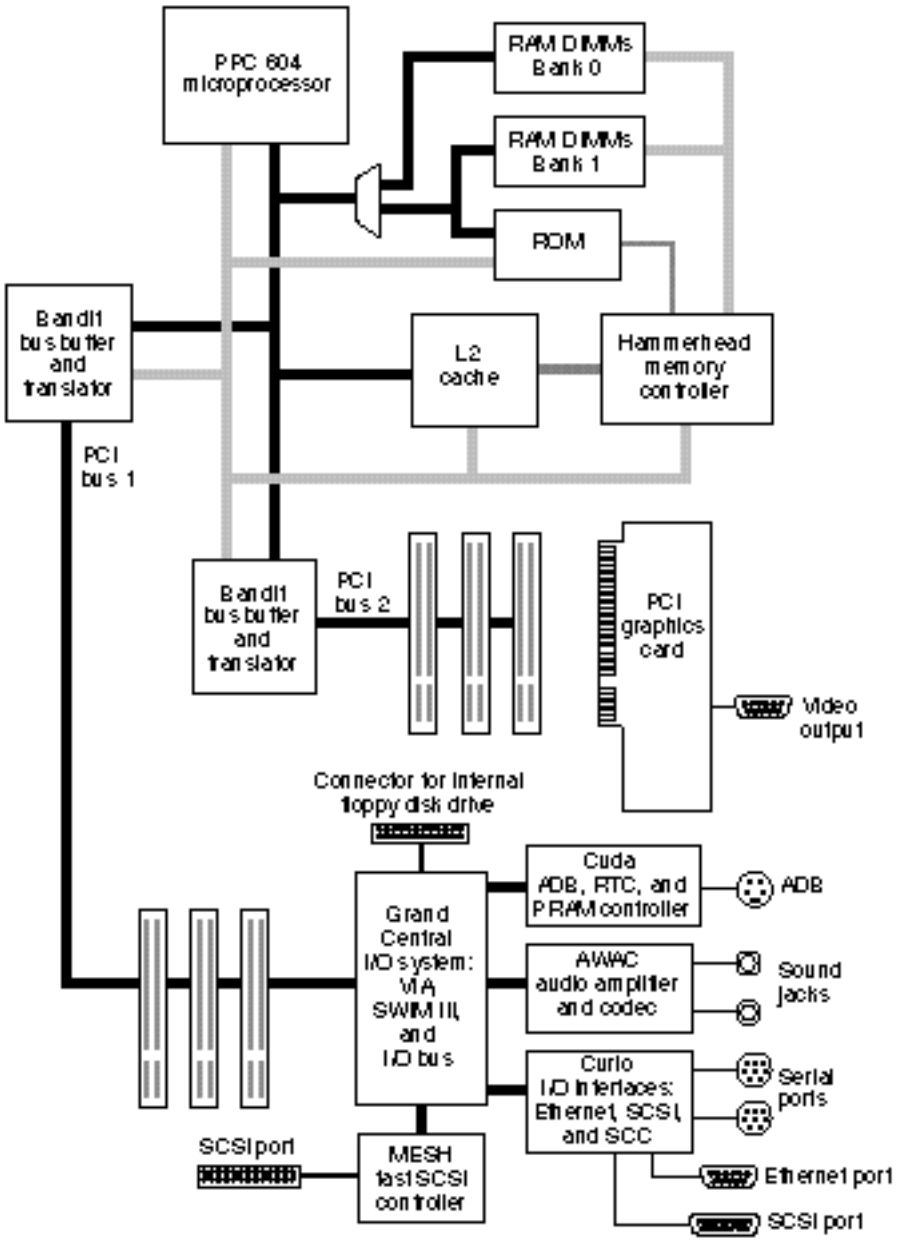
```
FF832990:   /gc@10
FF832DC8:     /53c94@10000
FF834650:       /sd@0,0
FF835280:       /st@0,0
FF835EF8:     /mace@11000
FF836D70:     /escc@13000
FF836EC8:       /ch-a@13020
FF837578:       /ch-b@13000
FF837C28:     /awacs@14000
FF837D10:     /swim3@15000
FF838E18:     /via-cuda@16000
FF8399A8:       /adb@0,0
FF839A98:         /keyboard@0,0
FF83A1E8:         /mouse@1,0
FF83A298:       /pram@0,0
FF83A348:       /rtc@0,0
FF83A810:       /power-mgt@0,0
FF83A930:     /mesh@18000
FF83C498:       /sd@0,0
FF83D0C8:       /st@0,0
FF83DDD0:     /nvram@1D000
FF83FB70:   /pci106b,1@B
FF83FD48:   /ATY,XCLAIM@D
FF848968:   /wayne.device@E


FF8494D8:   /wayne.device@F
FF83DF68: /bandit@F4000000
FF84A190:   /pci106b,1@B
FF84A368:   /pci1234,5678@D
FF84A670:   /pci1000,3@E
FF84A908:   /TRUV,TARGA2000PCI@F
FF83F1C0: /hammerhead@F8000000
ok
0 >
```

The first thing to note is that the device tree listing above is a description of the Macintosh 9500 block diagram. Compare the device tree to Figure 1. The device tree also has nodes, such as /chosen@0 and /packages@0 that may not be familiar and are not part of the block diagram. These are utility nodes that supply services. These nodes are described later. Look at the top node in the tree, which is /PowerPC,604@0.

This is a full path name and has a format as: /node-name0/node-name1/ ... /node-nameN. A node name has the following format. node-name@unit-address.

So the / in /PowerPC,604@0 is the AR bus. That's the gray bus in Figure 1, while the black bus is the PCI bus.

**Figure 1** *Power Macintosh 9500 Block Diagram*



## Properties and Methods of the Node /

Let's look at the properties and methods (i.e., words) of the root node /.

Note: When you first enter the user interface and before you enter anything, if you enter <words> you will see the entire list of words, which is too long to put in this note. After you select a device or make a node active, you will only see the words for that node. You must enter
< unselect-dev words > to see the full list, or of course enter < words > at the first input.

```
0 > pwd / ok
0 > .properties
name                    device-tree
model                   Power Macintosh
compatible              AAPL,9500
MacRISC
AAPL,cpu-id             3900A69D
#address-cells          00000001
#size-cells             00000001
clock-frequency         02FAF080

  ok
0 > words
dma-sync       dma-map-out     dma-map-in      dma-free        dma-alloc
map-out map-in    decode-unit
close           open
  ok
0 >
```

First, I displayed the device-path which made that node the active package, then displayed the properties, and then the words. It is interesting that the root node is / but the name property has a value of device-tree. This is the only case where these names differ. All other nodes will have their node-name and name property values equal.

Look at the value of the clock-frequency, which is 0x02FAF080. That value is 50 MHz which is the frequency of the AR bus.

Two other interesting properties are #address-cells and #size-cells, both of which have a value of 1. This indicates that the package is a bus node because only bus nodes have these properties. The value indicates the number of cells needed to define a physical address. In fact, #address-cells only applies to packages that define a physical address space such as the AR and PCI buses. #size-cells defines the size of an address which is, in this case, 32-bits. Later, we will see that the PCI bus requires larger values for these two properties.

Most of the words for this package deal with address mappings from virtual to physical. decode-unit is an interesting one. Look at the following code and remember it, because I will return to it when we look at the Bandit package.

```
0 > " 3456" decode-unit .s 3456
  ok
decode-unit takes a character string or an address and length from the stack
and returns a physical address. In this case, both values are the same since
the #address-size is 1.  That is, " 3456" is the same as the returned
physical address 3456.
```

Section 3.5 of the IEEE Specification names the following standard nodes that a device tree must support.

/

/aliases

/chosen

/openprom

/options

/packages

## Other Properties and Words

We have already looked at /. Here are the properties and words of the remainder.

```
0 > dev /aliases  ok
0 > .properties
name                    aliases
vci0                    2F636861 6F734046 30303030 30303000
pci1                    2F62616E 64697440 46323030 30303030 00
pci2                    2F62616E 64697440 46343030 30303030 00
fd                      2F62616E 6469742F 67632F73 77696D33 00
kbd                     2F62616E 6469742F 67632F76 69612D63
7564612F 6164622F 6B6
57962 6F617264

                        00
ttya                    2F62616E 6469742F 67632F65 7363632F
63682D61 00
ttyb                    2F62616E 6469742F 67632F65 7363632F
63682D62 00
enet                    2F62616E 6469742F 67632F6D 61636500
scsi                    2F62616E 6469742F 67632F35 33633934 00
scsi-int                2F62616E 6469742F 67632F6D 65736800

  ok
0 > words

  ok
0 > devalias
vci0                /chaos@F0000000
```

```
pci1                /bandit@F2000000
pci2                /bandit@F4000000
fd                  /bandit/gc/swim3
kbd                 /bandit/gc/via-cuda/adb/keyboard
ttya                /bandit/gc/escc/ch-a
ttyb                /bandit/gc/escc/ch-b
enet                /bandit/gc/mace
scsi                /bandit/gc/53c94
scsi-int            /bandit/gc/mesh
  ok
0 >
```

The aliases properties are hexadecimal values and not human-readable. Also, there are no words, but wait, there's more. Look at the devalias command and what it displayed. This is a list of aliases, in human readable form, to use for long path names.

Here are two separate ways to make the node called /pci1234,5678@D the active package.

```
0 > dev /bandit@F4000000/pci1234,5678  ok
0 > pwd /bandit@F4000000/pci1234,5678@D ok
0 > dev pci2/@d  ok
0 > pwd /bandit@F4000000/pci1234,5678@D ok
0 >
```

Look at the two differences in the dev command arguments. The first dev used the full path name and anyone who has made a typing error on a command line, knows that the interface is not very forgiving. So try using the second method, since it's equivalent and easier. Also note that the unit-address, in the case of the second dev command, was used instead of the node-name.

## The /openprom Node

Another standard system node is /openprom. Take a look because this node is the ID of the version of Open Firmware on your machine and should be used when submitting Open Firmware questions to Apple Developer Technical Support (DTS).

```
0 > dev /openprom .properties
name                    openprom
model                   Open Firmware, 1.0.5
relative-addressing

  ok
0 >
```

## Other Nodes Attached to the AR Bus

The remaining standard system nodes are detailed in the Specification and will not be covered here, so let's move to the three remaining nodes attached to the AR bus. These are the two Bandit ASICs and Hammerhead the memory controller. Here's hammerhead.

```
0 > dev /hammerhead  ok
0 > .properties
name                    hammerhead
reg                     F8000000  00000800
model                   AAPL,343S1142

  ok
0 > words

  ok
0 >
```

Not very interesting except for the reg property. This property along with its side-kick property called assigned-addresses will not be covered in detail in this Technote, but are the only subject of the next Technote in this series, *Technote 1044 - PCI Expansion ROM Contents for Mac OS 8, Part III in the Open Firmware Technote Series*.

## The PCI Bus

Now let's cross over a bridge chip to get to the PCI bus. Enter this for the first bus.

```
0 > dev pci1  ok
0 > .properties
name                    bandit
device_type             pci
model                   AAPL,343S1126
AAPL,interrupts         00000016
reg                     F2000000  02000000
#address-cells          00000003
#size-cells             00000002
clock-frequency         01FCA055
slot-names              0000E000 41310042 31004331 00
ranges     02000000 00000000 F3000000  F3000000  00000000 01000000
               01000000 00000000 00000000  F2000000  00000000 00800000
               02000000 00000000 80000000  80000000  00000000 10000000
bus-range               00000000 00000000

  ok
```

Now cross over to the second bus by entering this.

```
0 > dev pci2 .properties
name                    bandit
device_type             pci
model                   AAPL,343S1126
AAPL,interrupts         0000001A
reg                     F4000000  02000000
#address-cells          00000003
#size-cells             00000002
clock-frequency         01FCA055
slot-names              0000E000 44320045 32004632 00
ranges   02000000 00000000 F5000000 F5000000  00000000 01000000
             01000000 00000000 00000000 F4000000  00000000 00800000
             02000000 00000000 90000000 90000000  00000000 10000000
bus-range               00000001 00000001

  ok
0 >
```

Let's take a close look at these two nodes, since they are the same ASIC. Well, their names are the same, so how are they distinguished? By using the unit address. Look back at the devalias command. The first Bandit has a unit address of F2000000 while the second has F4000000. Device type and model are the same but the AAPL,interrupts properties are different.

AAPL,interrupts is not mentioned in any documentation and is a private property used to identify one Bandit chip interrupts from the other.

Clock-frequency is 1FCA055 hex which is 33 MHz.

Slot-names is interesting, since by definition it contains the slots used by Bandit and the labels. However, it looks like hex numbers, so let's decode this property.

```
0 > " pci1" select-dev  ok
0 > " slot-names" get-my-property  ok
3 > drop \ don't need Boolean  ok
2 > decode-int . E000  ok
2 > decode-string type A1 ok
2 > decode-string type B1 ok
2 > decode-string type C1 ok
2 > 2drop  ok
0 >
```

I first selected the device and then put the property on the stack. The first item wasn't needed. Look at the stack notation for get-my-property for details. The next item is an bit mask and as such can be rewritten as follows.

1110 0000 0000 0000 = E000

The LSB is 0 and so are all the others until you get to bits D, E, and F which are the unit addresses for the three slots in this Bandit chip. If you decode the other Bandit chip you get this.

```
0 > " pci2" select-dev  ok
0 > " slot-names" get-my-property drop  ok
2 > decode-int . E000  ok
2 > decode-string type D2 ok
2 > decode-string type E2 ok
2 > decode-string type F2 ok
2 > 2drop  ok
0 >
```

Well, the unit addresses are the same, but as you might expect, the labels are different. So this is how to locate the unit address associated with the slot that holds your device.

And finally, for the Bandit chip let's explore three properties that deal with address space. Actually, I'll touch on it here, but we go into more detail in the next Technote, *Technote 1044 - PCI Expansion ROM Contents for Mac OS 8, Part III in the Open Firmware Technote Series*. Here they are.

```
#address-cells          00000003
#size-cells             00000002
ranges          02000000 00000000 F5000000 F5000000  00000000 01000000
            01000000 00000000 00000000 F4000000  00000000 00800000
            02000000 00000000 90000000 90000000  00000000 10000000
```

Remember, Bandit is a child to the AR bus and a parent to the PCI bus. #address-cells and #size-cells is fixed at 3 and 2, respectively, for PCI but 1 and 1 for AR. That is how you read the ranges property. Let's break up the first entry to make that clearer.

(02000000 00000000 F5000000 for PCI) (F5000000 for AR)  (00000000 01000000 for size)

And this property is encoded as follows. (child-phys parent-phys size)

Look at the PCI Bus binding to IEEE specification for details of phys.hi cell, but here is part of it for convenience:

```
                Bit#: 33222222 22221111 11111100 00000000
                      10987654 32109876 54321098 76543210
phys.hi cell:         npt000ss bbbbbbbb dddddfff rrrrrrrr
phys.mid cell:        hhhhhhhh hhhhhhhh hhhhhhhh hhhhhhhh
phys.lo cell:         llllllll llllllll llllllll llllllll
```

Note bits 24 to 31. These bits show that the address spaces requested are relocatable. Two entries in the range property are for memory space and one is for I/O space. Well, that about covers the device tree down to but not including expansions slots. And as you might expect, your Macintosh does not have the same set of devices in its expansions slots that mine does.

## A Look at the ATI Card: Its Properties & Methods

Let's take a final look at the ATI card, which you might have. For those of you who don't have this card, here are its properties and methods.

```
0 > dev pci1/@d pwd .properties words
/bandit@F2000000/ATY,XCLAIM@D
vendor-id               00001002
device-id               00004758
revision-id             00000002
class-code              00030000
interrupts              00000001
min-grant               00000000
max-latency             00000000
devsel-speed            00000001
AAPL,interrupts         00000017
AAPL,slot-name          A1
fcode-rom-offset        00000000
name                    ATY,XCLAIM
model                   ATY,88800GX
device_type             display
character-set           49534F38 3835392D 3100
ATY,Rom#                3131332D 33333230 302D3131 3000
ATY,Mem#                3130302D 33313630 322D3030 00
ATY,Card#               3130392D 33333230 302D3030 00
ATY,Flags               00000000
reg                     00006800 00000000 00000000  00000000
00000000
                        02006810 00000000 00000000  00000000
01000000
width                   00000280
height                  000001E0
depth                   00000008
linebytes               00000280
iso6429-1983-colors
driver,AAPL,MacOS,PowerPC
                        4A6F7921 70656666 70777063 00000001 AB9350D5
00000000 00000000 00000000
                        00030002 00000000 FFFFFFFF 00000000 00005EF4
00005EF4 00005EF4 00000480
                        00040400 FFFFFFFF 00000000 000013D0 00000FC8
00000CE3 00006380 02010400
                        FFFFFFFF 00000000 00000000 00000000 000003F4
00000080 04040400 7C0802A6
```

```
                        FFFFFFFF 00000000 FFFFFFFF 00000000 FFFFFFFF
00000000 00000004 0000001C
                        00000001 00000114 0000012C 000003D0 00000001
00000002 00000000 00000000
                        00000000 00000009 00000000 00000000 00000010
00000000 00000000 0000000D
                        00000009 00000000 00000022 00000000 00000000
00000002 00000016 00000000
                    ... 00007063 bytes total
power-consumption       007270E0 007270E0
assigned-addresses      82006810 00000000 81000000  00000000
01000000


close                   restore          draw-logo             write
open                    read-rectangle
fill-rectangle               draw-rectangle  color@          color!
get-colors          set-colors

  ok
0 >
```

## What's Displayed

O.K., let's see what is displayed. Look at the properties vendor-id and device-id, then look at the name property. Notice that they are not equal. That is, the name property is ATY,XCLAIM, not pci1002,4758. This implies two things. The first is that the probing process did not have to declare a name for this device and that a name property was found in the expansion ROM by the probing process.

Also note that there is a property called fcode-rom-offset. It tells you that there is FCode in the ROM. Its value is zero, which means there is no offset in the ROM where the FCode resides. It's at the beginning of the ROM.

Look at the property called device-type. Its value is display, which is defined in the PCI Local Bus Specification. The device-type property is part of the declaration for this device as is the class-code property. Its value is 0x00030000. From Appendix D of the specification, we see that this class is for display controllers.

## Width and Height Properties

Let's look at two more properties called width and height. These show that the boot driver (i.e., Open Firmware) is set for a 13" monitor. Huh, you say? Well, 280 hex is 640 decimal while 1E0 hex is 480 and that is a 13" monitor.

Now let's gist the words (methods). These are the words required for a display boot driver. *Technote 1044 - PCI Expansion ROM Contents for Mac OS 8, Part III in the Open Firmware Technote Series* has the details, if you would like to go beyond this

Note.

```
0>words

close       restore      draw-logo     write       open        read-rectangle

fill-rectangle draw-rectangle color@      color!      get-colors    set-
colors
```

# Summary

This Note is intended to get you started with the device tree. *Technote 1061 — Fundamentals of Open Firmware, Part I: The User Interface* gave you an overview of using Forth and the Open Firmware user interface. The next Note in this series, *Technote 1044 – Understanding PCI Expansion ROM Choices for Mac OS 8, Part III in the Open Firmware Technote Series*, provides you with a detailed look at generating or not generating expansion ROM properties for memory allocation. It explains the differences in the assigned-addresses property when the probing process generates your reg and assigned-addresses property due to not finding a name and reg property in your expansion ROM.


## *Acknowledgments*

Thanks to Monte Benaresh, Ron Hochsprung, Jim Huffman, Pradeep Kathail, Holly Knight, Tom Maremaa, and Samuel Yan.