Device tree

Hardware Description

vs

Configuration

vs

Policy

Frank Rowand, Sony

November 2, 2016

161029_2114

My elevator pitch description

The purpose of device tree is to

describe non-discoverable hardware

ePAPR is vague

1.1 Purpose and Scope

...

This specification,..., provides a complete boot program to client program interface definition, ...

2.1 Overview

The ePAPR specifies a construct called a device tree to describe system hardware.

...

A device tree is a tree data structure with nodes that describe the devices in a system.

...

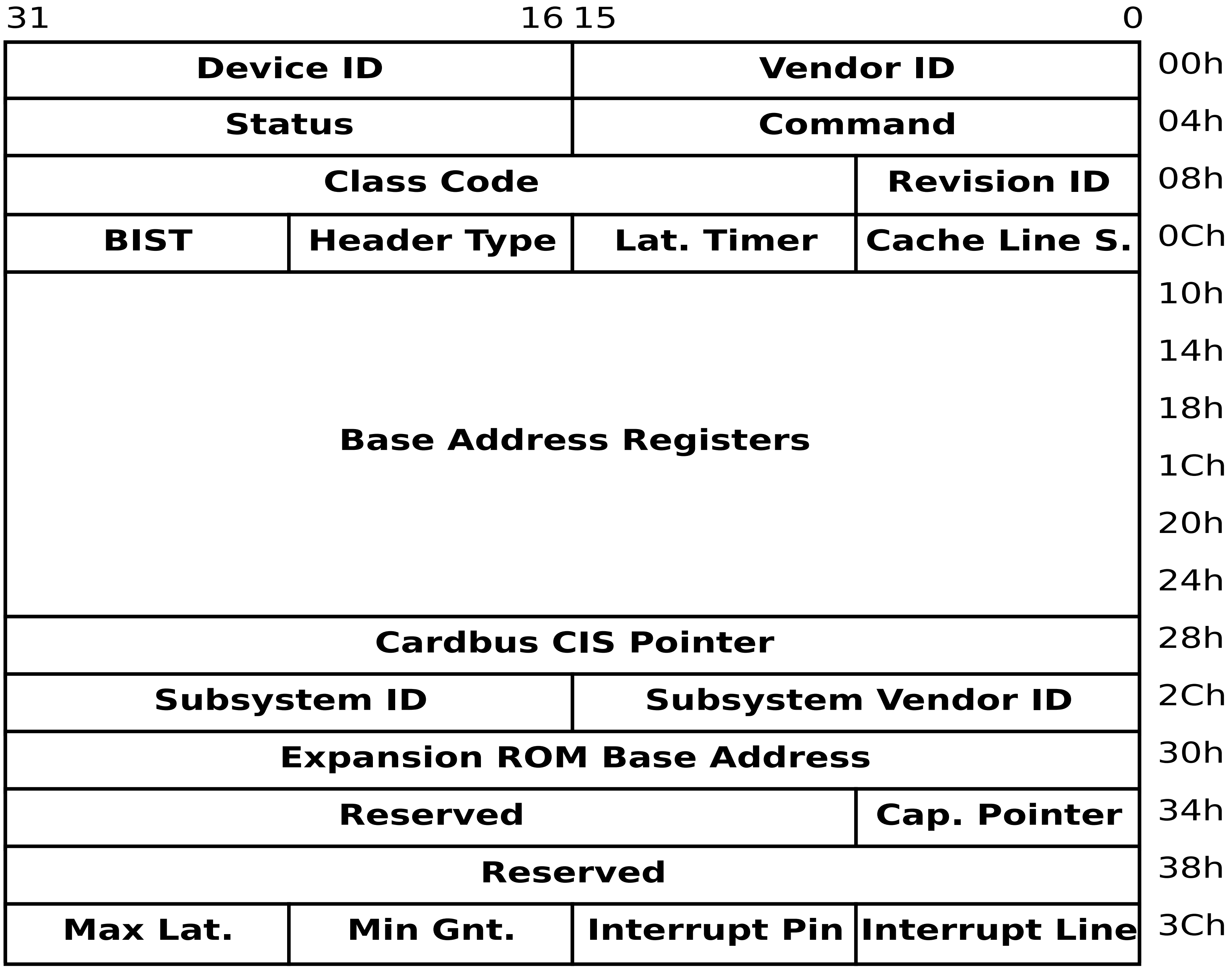
An ePAPR-compliant device tree describes device information in a system that cannot necessarily be dynamically detected by a client program.

my elevator pitch...

Implies information about vendor ID, device ID, address, connections within the system fabric (eg interrupts)

Somewhat like a PCI header..

PCI header



31	16	15	0	
Device ID		Vendor ID		00h
Status		Command		04h
Class Code			Revision ID	08h
BIST	Header Type	Lat. Timer	Cache Line S.	0Ch
Base Address Registers				10h
				14h
				18h
				1Ch
				20h
				24h
Cardbus CIS Pointer				28h
Subsystem ID		Subsystem Vendor ID		2Ch
Expansion ROM Base Address				30h
Reserved			Cap. Pointer	34h
Reserved				38h
Max Lat.	Min Gnt.	Interrupt Pin	Interrupt Line	3Ch

Obviously the world has moved on from the simple world of PCI headers

eg ACPI (?) and the modern device tree

What does the Linux world say?

What the device tree contains

Standard answer:

A description of the hardware

What the device tree contains

Real world:

A description of the hardware
and some other stuff

Other Stuff - trivial example

ePAPR section 3.5: Chosen

> “The chosen node does not represent a real device in the system but describes parameters chosen or specified by the system firmware at run time.”

Even the specification says there is other stuff.

Agenda

- what I would like discussed
- some vague definitions (a partial list)
- examples - quickly (can come back to later if time)
- discussion

What I would like discussed

Should dt contain things other than hw description?

- If yes, then what?
  - Guidance for proposing and accepting?
- If no, then where do these things belong?

Definitions of types of information

Intentionally vague

- We could spend hours debating the definitions
- Let's try to avoid that except where it is key to the discussion
- Exactly what category a piece of information fits into is not always black and white

Definitions - Hardware Description

Where the hardware is
What the hardware is
What the hardware is capable of doing

Examples:

- located at address 0x10000 - 0x12fff
- UART supports baud rates up to 38400
- which interrupt line the device is connected to
- a set of pins which could be MUXed to either an i2c bus controller or a UART is physically routed out to a UART connector

Definitions - Configuration

How the hardware should be set up, how drivers should behave, tuning parameters

Examples:
- set UART baud rate to 9600
- set a MUX to use a set of pins as an i2c bus
- set a MUX to use a set of pins as a UART

Definitions - Policy

How to do something, tuning parameters

Sometimes hard to distinguish between policy and configuration.

Examples:
- Set clock frequency for highest performance
- Set clock frequency for max battery duration

Definitions - Communications Channel

Communication Channel

A mechanism to transfer opaque data.

Example: the binary blob to program an FPGA

Special case (not opaque):
- “chosen” node
- communication from boot loader to kernel

Definitions - State

A property whose value reports current state.

Value of property may change over time
in the live tree.

Usually the device tree is used to convey a description of hw -- a one time event.
Drivers and subsystems maintain state in their own data structures.

Example:

- property “status”

Still looking for more examples

Definitions - Descriptive Information

Usually for human consumption

ePAPR section 6.1.2.3 label
  Property: label
  Value type: <string>
  Description:
  The label property defines a human readable string describing a device. The binding for a given device specifies the exact meaning of the property for that device.

Example:
  - the name of the UART beside the connector

Example - proposed

proscriptive -- Tells driver what to do

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
On some devices, CD is broken so that we must force the SDHCI into test
mode and set CD, so that it always detects an SD card as present.


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
+++ b/Documentation/devicetree/bindings/mmc/mmc.txt

+- force-sd-cd-test-mode: card detection is broken on device, force cd test
+  enable and cd test inserted so host will always detect a card.
```

Example - revised

describe hardware -- driver decides what to do

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Some board configurations can not support sd highspeed mode due to the
distance between the card slot and the controller. The card and controller
report that they are capable of highspeed however, so we need a
mechanism for specifying that the setup is incapable of supporting
highspeed mode.


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
+++ b/Documentation/devicetree/bindings/mmc/mmc.txt

+- sd-broken-highspeed: Highspeed is broken, even if the controller and card
+  themselves claim they support highspeed.
```

Example - mtd, in tree

how to probe for partition

```
commit 9d5da3a9b849
Date:   Tue Mar 9 12:27:56 2010 -0700

    mtd: extend physmap_of to let the device tree specify the parition probe

    This is to support custom partitioning schemes for embedded PPC. To use
    define your own mtd_part_parser and then add something like:
       linux,part-probe = "my_probe", "cmdlinepart";
    To the board's dts file.
```

Example - mtd, new solution

```
Subject: [PATCH 1/3] mtd: create a partition type device tree binding
Date: Thu, 29 Oct 2015 13:52:30 +0100

This solution is too Linux-specific and depend on some
Linux kernel-internal naming conventions. We need a proper
way of describing partition types that follow the pattern set by
other device tree bindings.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
+++ b/Documentation/devicetree/bindings/mtd/mtd-physmap.txt

+ - partition-type : a flash partition type to expect and probe for
+   on this device. See "partition.txt" for available partition types.
```

Example - configuration

in tree
Date: Tue, 24 Nov 2015

- -
This chip supports two power modes.
1. "one-shot" mode - the chip activates and executes one complete conversion loop and then shuts itself down. This is the default mode chosen for raw reads.
2. "continuous" mode - the chip takes continuous measurements.

Continuous mode is more expensive power-wise but may be more reliable.
Add a property so that if preferred, the default power mode for raw reads can be set to continuous.

Example - configuration

in tree
Date: Tue, 24 Nov 2015

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
+++ b/Documentation/devicetree/bindings/iio/light/us5182d.txt

+- upisemi,continuous: This chip has two power modes: one-shot (chip takes one
+                      measurement and then shuts itself down) and continuous (
+                      chip takes continuous measurements). The one-shot mode is
+                      more power-friendly but the continuous mode may be more
+                      reliable. If this property is specified the continuous
+                      mode will be used instead of the default one-shot one for
+                      raw reads.
```

Example - limit driver action

in tree

- -
Documentation/devicetree/bindings/regulator/regulator.txt

- regulator-always-on: boolean, regulator should never be disabled

- -
Not likely to be removed due to extensive use:

```
$ git grep "regulator-always-on" | grep ".dts[i]*" | wc -l
2217
```

Example - message to bootloader

in tree
Date: Wed, 6 Jul 2016
Property values are magic u32 constants

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
+++ b/Documentation/devicetree/bindings/power/reset/reboot-mode.txt

+All mode properties are vendor specific, it is a indication to tell
+the bootloader what to do when the system reboots, and should be named
+as mode-xxx = <magic> (xxx is mode name, magic should be a none-zero value).


+Example:
+  reboot-mode {
+     mode-normal = <BOOT_NORMAL>;
+     mode-recovery = <BOOT_RECOVERY>;
+     mode-bootloader = <BOOT_FASTBOOT>;
+     mode-loader = <BOOT_BL_DOWNLOAD>;
+  }
```

Example - message to bootloader

- -

An analogy could be what values to write to a command register:

  latch address from data register

  read data from latched address

  write data to latched address

Example - configuration

Date: Tue, 23 Feb 2016

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
+++ b/Documentation/devicetree/bindings/arm/l2c2x0.txt

+- arm,dynamic-clock-gating : L2 dynamic clock gating. Value: <0> (forcibly
+  disable), <1> (forcibly enable), property absent (retain firmware settings)
+- arm,standby-mode: L2 standby mode enable. Value <0> (forcibly disable),
+  <1> (forcibly enable), property absent (retain firmware settings)


- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

Note: these two properties are consistent with a long list of similar properties in this binding.

Example - configuration

Proposed

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
+++ b/Documentation/devicetree/bindings/net/wireless/ieee80211.txt

+The following properties are common to the IEEE 802.11 controllers:
+
+- enable-ieee80211-2ghz: indicates that the 2.4GHz frequency band should be
+  enabled for this device. This must not be used together with the
+  "disable-ieee80211-2ghz" at the same time.
+- enable-ieee80211-5ghz: indicates that the 5GHz frequency band should be
+  enabled for this device. This must not be used together with the
+  "disable-ieee80211-5ghz" at the same time.
+- disable-ieee80211-2ghz: indicates that the 2.4GHz frequency band should be
+  disabled for this device.
+- disable-ieee80211-5ghz: indicates that the 5GHz frequency band should be
+  disabled for this device
```

Example - configuration data

Proposed alternative for previous slide

- -

If EEPROM is commonly wrong or missing, then seems like you want to plan ahead and support both enable and disable.

The other approach I've mentioned before is just put raw EEPROM data into DT to override the EEPROM. This would be better if there's a large number of settings you need.

Discussion

What I would like discussed

Should dt contain things other than hw description?

- If yes, then what?
    - Guidance for proposing and accepting?
- If no, then where do these things belong?