

Unified Device Properties Interface for ACPI and Device Trees

Rafael J. Wysocki

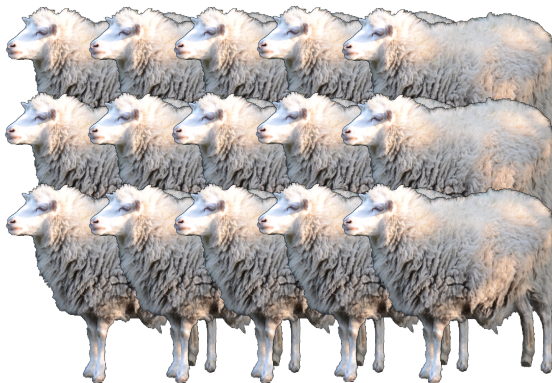
Intel Open Source Technology Center

February 19, 2015

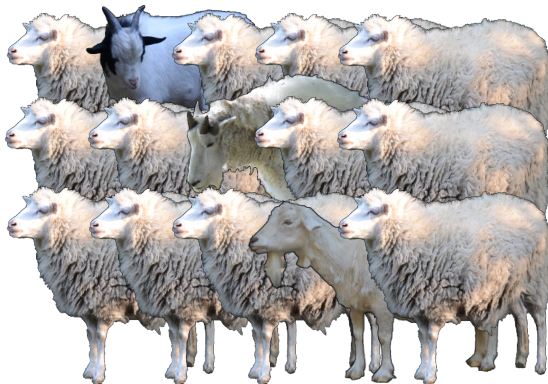
Outline

- 1 Introduction
 - Origins of ACPI
 - Motivation for Device Trees
- 2 ACPI vs Device Trees
 - Differences and Analogies
 - Putting It All Together
- 3 Unification of API for Device Drivers
 - Way to Go
 - Unified Device Properties API
- 4 Resources

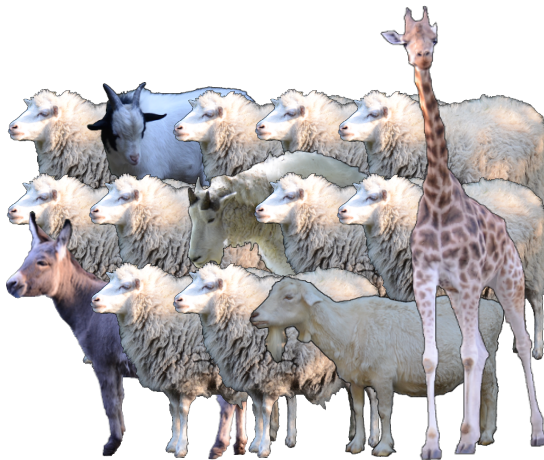
PC Monoculture (Hardware Compatibility Everywhere)



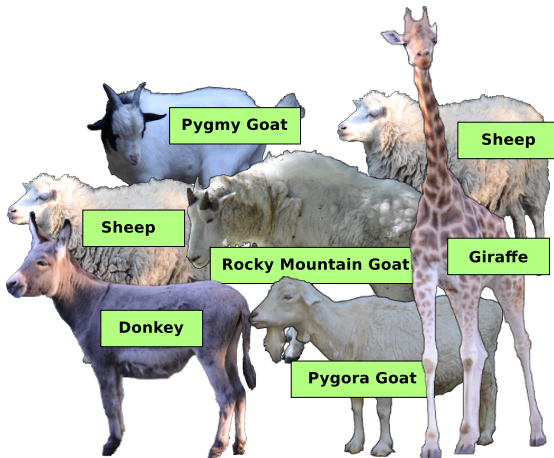
Differences Appear (Slightly Incompatible Hardware)



Substantial Differences (Not Really Compatible Any More)

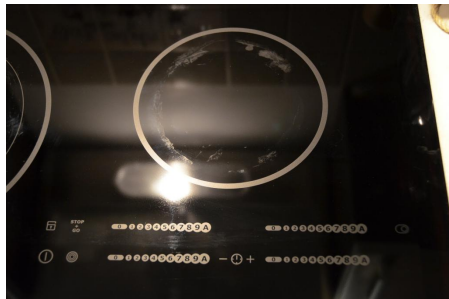


Solution: Platform Description Protocol

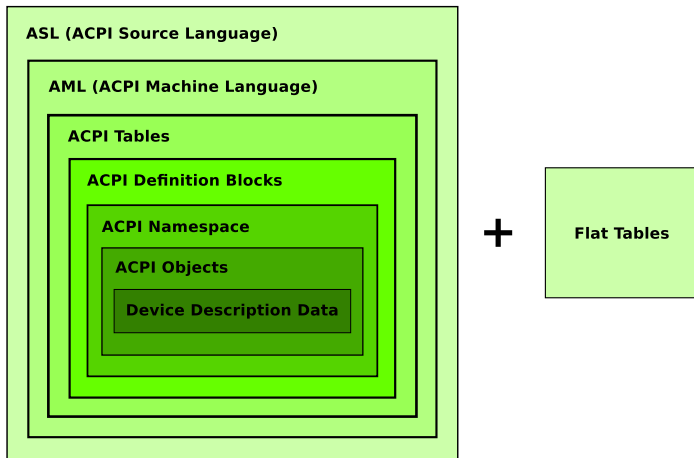


ACPI: Advanced Configuration and Power Interface

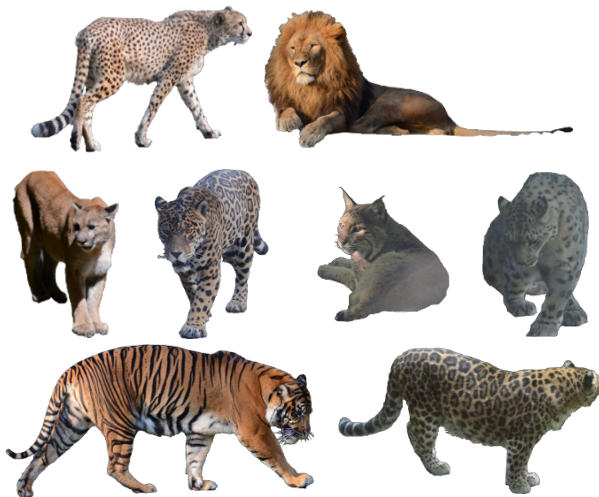
Platform description + high-level interface for driving platform hardware.



ACPI Complexity



ARM: Same Architecture, Different Platforms



Intel
OpenSource
TECHNOLOGY CENTER

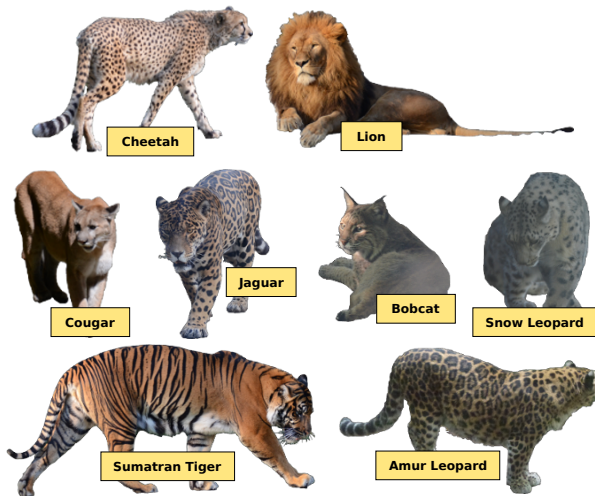
ARM: Lessons Learned

- ① Code duplication avoidance is a priority.
- ② Custom per-platform kernel builds are a bad idea.

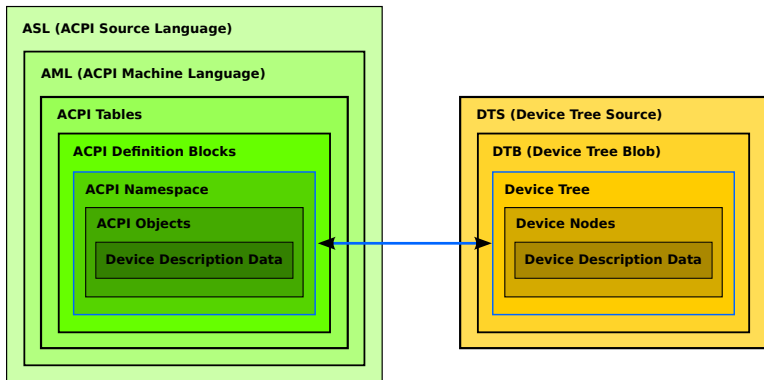


Intel
OpenSource
TECHNOLOGY CENTER

Solution: Platform Description Protocol



ACPI and DT Side-by-Side



DT, ACPI and Device Drivers

Tipping point (ca. 2012)

- x86 processors become SoCs (and include ARMish IP blocks).
- Server vendors want to use ACPI with ARM64.

Problems

- Some drivers have grown dependencies on DT (or ACPI).
- Until recently ACPI was missing a way to operate properties.

Flattened Device Tree

```
/ {  
    node1 {  
        a-string-property = "A string";  
        a-string-list-property = "first string", "second string";  
        a-cell-property = <1 2 3 4>; /* each number (cell) is a uint32 */  
        child-node1 {  
            first-child-property;  
            second-child-property = <1>;  
            a-string-property = "Hello, world";  
        };  
        child-node2 {  
        };  
    };  
    node2 {  
        an-empty-property;  
        a-byte-data-property = [0x01 0x23 0x34 0x56];  
        child-node1 {  
        };  
    };  
};
```

The Missing Link

_DSD (Device Specific Data) introduced in ACPI 5.1

```
Name (_DSD, Package () {  
    ToUUID("<UUID string>"), // Format identifier  
    Package () {  
        ... // Device data in the given format  
    }  
    ...  
})
```

Device Properties UUID: daffd814-6eba-4d8c-8a91-bc9bbf4aa301

```
Name (_DSD, Package () {  
    ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),  
    Package () {  
        Package {"a-string-property", "A string"},  
        Package {"a-string-list-property", Package {"first string", "second string"}};  
        Package {"a-cell-property", Package {1, 2, 3, 4}};  
    }  
})
```

Driver Adaptation: Simple Case

```
- if (of_property_read_u32(np, "size", &val) == 0 ||
-     of_property_read_u32(np, "at25,byte-len", &val) == 0) {
+ if (device_property_read_u32(dev, "size", &val) == 0 ||
+     device_property_read_u32(dev, "at25,byte-len", &val) == 0) {
    chip->byte_len = val;
} else {
    dev_err(dev, "Error: missing \"size\" property\n");
    return -ENODEV;
}

- if (of_property_read_u32(np, "pagesize", &val) == 0 ||
-     of_property_read_u32(np, "at25,page-size", &val) == 0) {
+ if (device_property_read_u32(dev, "pagesize", &val) == 0 ||
+     device_property_read_u32(dev, "at25,page-size", &val) == 0) {
    chip->page_size = (u16)val;
} else {
    dev_err(dev, "Error: missing \"pagesize\" property\n");
    return -ENODEV;
}

- if (of_property_read_u32(np, "at25,addr-mode", &val) == 0) {
+ if (device_property_read_u32(dev, "at25,addr-mode", &val) == 0) {
    chip->flags = (u16)val;
} else {
```

Problem: Abstract Device Objects

Device objects (in the firmware) without struct device representation.

```
enum fwnode_type {
    FWNODE_INVALID = 0,
    FWNODE_OF,
    FWNODE_ACPI,
};

struct fwnode_handle {
    enum fwnode_type type;
};

bool is_of_node(struct fwnode_handle *fwnode);
struct device_node *of_node(struct fwnode_handle *fwnode);

bool is_acpi_node(struct fwnode_handle *fwnode);
struct acpi_device *acpi_node(struct fwnode_handle *fwnode);
```

API Overview

```
bool fwnode_property_present()
bool fwnode_property_read_bool()
```

```
int fwnode_property_read_u8()
int fwnode_property_read_u16()
int fwnode_property_read_u32()
int fwnode_property_read_u64()
int fwnode_property_read_string()
```

```
int fwnode_property_read_u8_array()
int fwnode_property_read_u16_array()
int fwnode_property_read_u32_array()
int fwnode_property_read_u64_array()
int fwnode_property_read_string_array()
```

```
struct fwnode_handle *device_get_next_child_node()
void fwnode_handle_put()
```

```
device_for_each_child_node(dev, child)
```

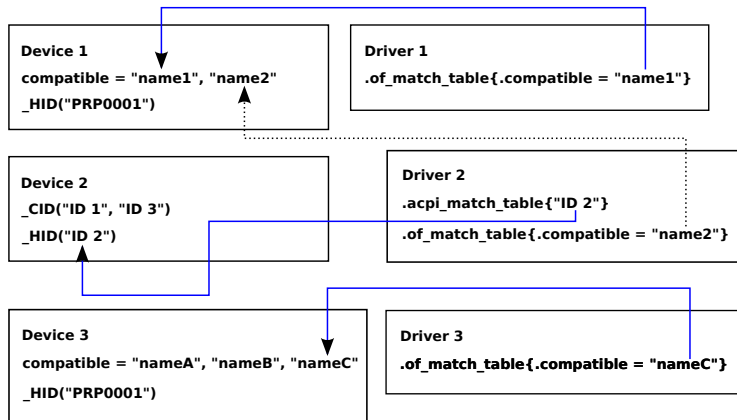
```
unsigned int device_get_child_node_count()
```

```
bool device_property_present()
bool device_property_read_bool()
```

```
int device_property_read_u8()
int device_property_read_u16()
int device_property_read_u32()
int device_property_read_u64()
int device_property_read_string()
```

```
int device_property_read_u8_array()
int device_property_read_u16_array()
int device_property_read_u32_array()
int device_property_read_u64_array()
int device_property_read_string_array()
```

Binding Drivers to Devices



Conclusion

- The unified device properties API is in the mainline (3.19).
- New drivers should try to use it (instead of the plain DT API).

References



UEFI, *Advanced Configuration and Power Interface Specification, Revision 5.1*
(http://www.uefi.org/sites/default/files/resources/ACPI_5_1release.pdf).



UEFI, *Device Properties UUID For _DSD, Revision 1.0*
(http://www.uefi.org/sites/default/files/resources/_DSD-device-properties-UUID.pdf).



Device Tree Wiki, *Device Tree Usage* (http://www.devicetree.org/Device_Tree_Usage).



Thomas Petazzoni, *Device Tree for Dummies*
(<http://events.linuxfoundation.org/sites/events/files/slides/petazzoni-device-tree-dummies.pdf>).

Documentation And Source Code

- Documentation/devicetree/
- Documentation/acpi/
- drivers/acpi/property.c
- include/linux/property.h
- drivers/acpi/
- drivers/of/
- drivers/leds/leds-gpio.c
- drivers/input/keyboard/gpio_keys_polled.c
- drivers/misc/eeprom/at25.c

Legal Information

Intel is a trademark of Intel Corporation in the U. S. and other countries.

*Other names and brands may be claimed as the property of others.

Copyright © 2014 Intel Corporation, All rights reserved.

Thanks!

Thank you for attention!