

Re: [PATCH 1/5] power: RFC: introduce a new power API

# Re: [PATCH 1/5] power: RFC: introduce a new power API

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-12/msg05664.html>

---

- *From:* Andres Salomon <[dilinger@xxxxxxxxxx](mailto:dilinger@xxxxxxxxxx)>
  - *Date:* Mon, 17 Dec 2007 02:41:39 -0500
- 

On Mon, 17 Dec 2007 08:51:23 +0300

Anton Vorontsov <[cbouatmailru@xxxxxxxxxx](mailto:cbouatmailru@xxxxxxxxxx)> wrote:

Hello Andres, David,

Firstly, Andres, thank you for the efforts.

I quite foreseen what exactly you had in mind when we were discussing the idea. With patches it's indeed easier to show flaws of this approach.

On Sun, Dec 16, 2007 at 09:36:24PM -0500, David Woodhouse wrote:

On Sun, 2007-12-16 at 21:24 -0500, Andres Salomon wrote:

This API has the power\_supply drivers device their own device\_attribute list; I find this to be a lot more flexible and cleaner.

I don't see how this is more flexible and cleaner. See below.

For example, rather than having a function with a huge switch statement (as `olpc_battery` currently has), we have separate callback functions.

Is this an improvement? Look into `ds2760_battery.c`. I scared to imagine what it will look like after conversion.

Why? It would not look bad after conversion. Basically:

```
static ssize_t ds2760_battery_get_status(struct device *dev,
```

Re: [PATCH 1/5] power: RFC: introduce a new power API

Re: [PATCH 1/5] power: RFC: introduce a new power API

```
struct device_attribute *attr, char *buf)
{
    struct ds2760_device_info *di = to_ds2760_device_info(psy);
    return power_supply_status_str(di->charge_status, buf);
}
static ssize_t ds2760_battery_get_voltage_now(struct device *dev,
struct device_attribute *attr, char *buf)
{
    struct ds2760_device_info *di = to_ds2760_device_info(psy);
    ds2760_battery_read_status(di);
    return sprintf(buf, "%d\n", di->voltage_uV);
}
```

...an so on.

If I wanted to get really clever, I could do:

```
#define DS2760_CALLBACK(name, fmt, var) \
static ssize_t ds2760_battery_get_##name(struct device *dev, \
struct device_attribute *attr, char *buf) \
{ \
    struct ds2760_device_info *di = to_ds2760_device_info(psy); \
    ds2760_battery_read_status(di); \
    return sprintf(buf, fmt, var); \
}
```

```
DS2760_CALLBACK(voltage_now, "%d\n", di->voltage_uV)
DS2760_CALLBACK(current_now, "%d\n", di->current_uA)
```

etc.. but, I'm not trying to compress lines of code, I'm trying to ensure things are readable.

As for olpc's "huge switch statement", it could be split into functions `_without_` drastic changes to PSY class. As the bonus, you will get `_inlining_` of these functions by gcc, because there will be just single user of these functions. With "exported-via-pointers" functions you can't do that.

You have tons of similar functions with similar functionality, that only differs by the data source. That scheme was in the early PSY class I posted here this summer. And I turned it down, fortunately.

On a bet, I can convert "huge switch statement" to nicely look switch statement. It will as nice as ds2760's.

The problem isn't in the PSY class.

Re: [PATCH 1/5] power: RFC: introduce a new power API

We're still going to be stuck with a huge switch statement. Yes, it wouldn't be *\*as\** big, but `ds2760_battery.c` has a decently sized switch statement, and `olpc_battery.c` has even more properties.

The huge switch statement is the least of my worries, though. Getting rid of it is just a bonus.

We're not limited  
to drivers only being able to pass 'int' and 'char\*'s in sysfs,

You're not limited to "int" and "char \*". Anything more than that is unnecessary, so far.

See below about the eeprom dump. Originally, it was desired for this to be a hex string; after that, binary. Of course, once I actually started adding `device_attributes` to `olpc_battery.c`, I started wondering; why not just make *\*all\** the properties `device_attributes`? And, what if I want to show something larger than a signed int? What if I have a value that I want to pad with 0's?

we're  
not forced to keep a global string around in memory (as is  
again the  
case for `olpc_battery`'s serial number code),

If battery chip can report strings, then you anyway must keep it in the memory. The question is when to allocate memory and when to free it. Side question is for how long to keep it.

Given that that string is small enough (dozen bytes), it's doesn't matter for how long we'll allocate it. So, in most cases it's easy to answer: allocate at probe, free at remove, so keep it for whole battery lifetime. (In contrast, adding tons of functions will waste *\_much more\_* space than these dozen bytes!)

IIRC this is the main difficulty you're facing with current properties approach. You've converted whole class to the something different.. but you didn't show a single user of that change. Sorry, `olpc` still using hard-coded manufacturer string:

Re: [PATCH 1/5] power: RFC: introduce a new power API

Well, no, I was talking about the serial number string. It's not upstream yet, it's just in OLPC's repo.

<http://dev.laptop.org/git?p=olpc-2.6;a=commitdiff;h=f9b4313060ab9047942707da6d3084f7792e714c>

Note bat\_serial, 17 bytes sitting around. That's actually not that bad, merely awkward. Worse (and what caused me to start reworking the API) was a dump of the EEPROM contents in hex; basically, (0x80-0x20)\*2. So, a buffer of 192 bytes was required to be kept around in memory. As we add more features, we end up w/ more and more wasted space.

```
+static ssize_t olpc_bat_manufacturer(struct device *dev,  
+ struct device_attribute *attr, char *buf)  
+{  
+ int ret;  
+ uint8_t ec_byte;  
+  
+ ret = olpc_bat_get_status(&ec_byte);  
+ if (ret)  
+ return ret;  
+  
+ ec_byte = BAT_ADDR_MFR_TYPE;  
+ ret = olpc_ec_cmd(EC_BAT_EEPROM, &ec_byte, 1, &ec_byte, 1);  
+ if (ret)  
+ return ret;  
  
+ switch (ec_byte >> 4) {  
+ case 1:  
+ strcpy(buf, "Gold Peak");  
break;  
+ case 2:  
+ strcpy(buf, "BYD");  
break;  
+ default:  
+ strcpy(buf, "Unknown");  
break;  
+ }  
+  
+ return ret;  
+}
```

In other words: all these strings can and should be static. Why spend cpu cycles on strcpy'ing things that can be not strcpy'ed?

Sure, except we have to spend the CPU cycles copying them into the buffer \*anyways\*. After all, that's how power\_supply\_sysfs gets the returned char\*'s into the sysfs buffer..

Re: [PATCH 1/5] power: RFC: introduce a new power API

I don't see S/N function. I'm sure it could be implemented easily using today's properties approach.

we don't have ordering  
restrictions w/ the return value being interpreted based upon  
where it's  
located in the array... etc.

What exact "restrictions" you're talking about? There are no restrictions per se.

The power\_supply\_properties enum; 'int's are listed first, 'char\*'s are listed second. power\_supply\_show\_property() assumes that anything after POWER\_SUPPLY\_PROP\_MODEL\_NAME in that list is a char\*, and anything before it is an integer. No other options are available, and if you accidentally put an int property somewhere in the strings section; well, too bad.

The other API seems to encourage driver authors to get their custom sysfs knobs into the list of sysfs knobs, and this one doesn't.

Yes, API is encouraging to add knobs, but not just any knobs. Only ones that make sense as a property of a PSY (opposite to some kind property of PSY driver itself). The count of such properties is limited, physically.

I'll reference

<http://dev.laptop.org/git?p=olpc-2.6;a=commitdiff:h=f9b4313060ab9047942707da6d3084f7792e714c>  
again, because it's a good example of the problems w/ the current API.

Basically, we need to add POWER\_SUPPLY\_PROP\_ACCUM\_CURRENT to the core driver. I don't see any other power\_supply driver actually needing that; however, the alternative is having magical #defines for those that are in olpc\_battery.c.. and we hope that the core enum list never accidentally clobbers our internal defines. ie,

```
#define OLPC_PROP_ACCUM_CURRENT 0x30
```

Re: [PATCH 1/5] power: RFC: introduce a new power API

Re: [PATCH 1/5] power: RFC: introduce a new power API

Inside of `olpc_bat_get_property`:

```
case OLPC_PROP_ACCUM_CURRENT:  
...  
break;
```

And, we hope that `include/linux/power_supply.h` never grows to include 48 entries inside the `power_supply_properties` enum (otherwise, we could end up with `power_supply_show_property()` mangling `OLPC_PROP_ACCUM_CURRENT`). Realistic? Not overly. But, having to choose arbitrary numbers is *\*not\** a desirable way to do things, nor is requiring every single property from every single driver to end up in the `power_supply_properties` enum list.

The `power_supply_class` docs say, "Power supply class is extensible, and allows to define drivers own attributes."; however, they don't say *\*how\** to define their own attributes, and I don't see any `power_supply` drivers that are doing that. This leads me to believe that the API is not suited for this sort of thing.

I'm recalling question about raw data. No, `PSY` class isn't for raw data you're getting from the firmware. Implement driver-specific binary attribute, that will contain device-specific raw data. Ideally, you should not export raw data at all (though, good idea is to export them into the debugfs).

Sure, debugfs is an option; however, for a quick and dirty tests, one of our hardware people wanted to be able to simply view the contents of the eeprom (and it wasn't clear what format was best. My assumption was that hex would be okay, but it turned out to be fairly unreadable. Binary piped to `od` was much nicer).

If there is interest in this API, I'll convert the rest of the `power_supply` drivers over to it and resubmit patches.

Looks sane enough to me.

Heh..

Re: [PATCH 1/5] power: RFC: introduce a new power API

Re: [PATCH 1/5] power: RFC: introduce a new power API

If Anton has no objections, I'll merge it.

Sorry, lots of objections.

Yes, the callback API is slightly more verbose. I see that as a good thing; I don't think the power\_supply core should be magically turning results into what it thinks they should be. I also think that moving code out of the power\_supply core, and into specific power\_supply drivers is a good idea. If I'm using pda\_power.ko, I don't want all the excess baggage that olpc\_battery.ko might need (and therefore had to include in power\_supply\_core).

Finally, to compare code size.. The current API:

```
-rw-r--r-- 1 dilinger dilinger 56768 2007-12-17 02:21 drivers/power/pda_power.o
-rw-r--r-- 1 dilinger dilinger 31830 2007-12-17 02:21 drivers/power/power_supply_core.o
-rw-r--r-- 1 dilinger dilinger 25120 2007-12-17 02:21 drivers/power/power_supply_leds.o
-rw-r--r-- 1 dilinger dilinger 30220 2007-12-17 02:21 drivers/power/power_supply_sysfs.o
```

The API I'm suggesting:

```
-rw-r--r-- 1 dilinger dilinger 55208 2007-12-17 02:19 drivers/power/pda_power.o
-rw-r--r-- 1 dilinger dilinger 30832 2007-12-17 02:19 drivers/power/power_supply_core.o
-rw-r--r-- 1 dilinger dilinger 24232 2007-12-17 02:19 drivers/power/power_supply_leds.o
-rw-r--r-- 1 dilinger dilinger 26680 2007-12-17 02:19 drivers/power/power_supply_sysfs.o
```

I think I can actually do even better, but I'm not going to bother unless folks like the new API.

--

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>