

Re: [PATCH] input: extend EV_LED

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-02/msg06429.html>

- *From:* Richard Purdie <rpurdie@xxxxxxxx>
 - *Date:* Sun, 18 Feb 2007 11:05:56 +0000
-

On Fri, 2007-02-16 at 01:12 -0200, Henrique de Moraes Holschuh wrote:

On Thu, 15 Feb 2007, Richard Purdie wrote:

This has been discussed in several places several times. The problem with hardware accelerated flashing is that you're often limited to certain constraints (this case being no exception) and indicating what these are to userspace in a generic fashion is difficult.

The ability to blinking at one rate is *very* common on laptops. Blinking at a few discrete rates is also common enough. They should be supported in a generic way.

I just said that finding a way to do it generically is difficult, not that we shouldn't do it.

I want to convert ibm-acpi to the led interface, but if it means I have to provide custom attributes on top of the led class, it sort of defeats most of the purpose of using the led class to begin with -- it will NOT be generic.

Even if half your functionality is exposed through the class, that half that is standardised rather than adhoc. Having said that, you shouldn't need any custom attributes though.

If I have to provide those attributes elsewhere in the sysfs tree other than somewhere in the led class, then it defeats the purpose of using the led class completely: I will just scrap the idea. I am not going to remove functionality. And I am not going to emulate in software something the hardware can do, especially when that means bothering the EC with a slow ACPI-subsystem-gated LPC bus IO port access for no good reason.

Here's a suggestion for a simple, non-overengineered interface: a "blink" attribute (on/off) for leds which can hardware-blink. Only one blink

Re: [PATCH] input: extend EV_LED

frequency is common enough that this attribute by itself is very useful (e.g. it is all a ThinkPad and most WiFi/network card leds need).

Right, but blinking is not an LED attribute but more of an action for the LED so what we need is an LED blink trigger. Rather than the timer trigger which takes a variety of options, this blink trigger could just take an on/off value. In the absence of hardware capability, we can emulate it. I like the idea of a simple blink trigger...

For hardware-blink leds with various frequencies, there is the typical way to provide such things: give us a RO blink_available_frequencies attribute which says which discrete frequencies are allowed (space separated), and a RW blink_frequency attribute to set the frequency. If instead of blink_available_frequencies, the driver provides RO blink_frequency_min and _max attributes, then it means it can blink on that range of freqs.

This is quite complex and whilst we could certainly have a trigger that did this, we already have a variable frequency trigger. See below.

That is simple enough to implement and use, and generic enough. You just need to set in stone if you want the freq in Hz, or a submultiple. You can even implement an optional "blink" software emulation that drivers can hook into for systems where the driver *knows* that led access is fast, but there is no hardware blinking emulation.

If a trigger/attribute appears for an LED, its behaviour needs to be the same for all LEDs.

One way I've come up with is adds capability to the class to have LED specific triggers and you can then expose these hardware capabilities as an extra trigger specific to the LED.

How would that look like? It doesn't sound too bad. Could you give us an example of what the tree would look like, and what the attributes would be (and do)?

Another proposal more specific to this use case was to have some information behind the scenes which the software timer based trigger could use to turn on the "hardware acceleration" if present and capable of the requested mode. This might just need a function pointer in the core so could be quite neat.

Re: [PATCH] input: extend EV_LED

This looks like a severely overengineered way to deal with the problem at first glance.

Which means you haven't thought about it as its quite simple in software terms. The LED driver can optionally implement a couple of functions:

```
set_blink(enum frequency)
set_blink_frequency(int delay_on, int delay_off)
```

These are not exported as an attribute directly and are just something triggers can use. Any trigger needing blinking behaviour calls one of these functions as appropriate and if implemented.

The enum reflects a spectrum of loosely defined frequencies, a bit like brightness maybe in a range 0–6. The idea is these are loose definitions and the driver will attempt a loose match, using any hardware blinking if available.

In the case of an LED with a full blown PWM capability (which can support near enough any frequency), it could just implement `set_blink_frequency()` and the LED core could provide a `set_blink()` function which translated into a call to `set_blink_frequency()` with some predefined frequency defaults. If it didn't support the parameters passed, it returns an error which the trigger would have to handle with a software default.

Yes, this is a slightly complicated solution but it solves 101 other scenarios other than just yours and allows hardware acceleration of things like the generic timer trigger as well as hardware acceleration of any trigger which wants a flashing LED instead of an LED thats just turned on.

Regards,

Richard

–

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

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

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