

Re: [PATCH] atmel_serial: update the powersave handler to match serial core

Re: [PATCH] atmel_serial: update the powersave handler to match serial core

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-09/msg06583.html>

- *From:* Anti Sullin <anti.sullin@xxxxxxxxxxxxxxxx>
 - *Date:* Sat, 20 Sep 2008 00:49:29 +0300
-

Haavard Skinnemoen wrote:

Michael Trimarchi <trimarchimichael@xxxxxxxx> wrote:

I agree it would be useful. It would require changing the port mux configuration from the driver though, and there's no standardized interface for doing that. Maybe this is a good motivation to come up with one?

I think that a driver can do the request to a the gpio layer (may can be implemented by the gpio-lib) and give it only the gpio. The "gpio-lib" can save and restore the status of the gpio, and request the handler, passing the gpio-id as a data. So when the handler fire, we can now which peripheral is interested on the wake-up event.

I don't think the gpio layer is supposed to touch the portmux. David has always been very clear about that. But if we somehow manage to get the pin configured as a GPIO, we can use the GPIO layer to request a pin change interrupt.

It *might* work even if you don't reconfigure the pin as a GPIO...but then I think we'd be relying on undocumented behaviour.

I believe that you don't need to reconfigure the pin. The gpio hw does not require the pin to be multiplexed to any given state, so does not request_irq (correct?). I did this trick in my [2/3] MCI driver patch I sent to arm-linux-kernel at 18.03.2008 (the e-mail subject line was wrong, [1/3], though) and I'm using that patch still on my production devices to avoid some rare but critical SD data corruption (mainline kernel still screws up the filesystem on SD sometimes). The same should be easily usable on serial port, too.

Re: [PATCH] atmel_serial: update the powersave handler to match serial core

Btw, I assume the first character you receive will be lost
when you do
this, right?

Yes, I haven't done a lot of test to see how many chars are lost (sure one I think). Depends on the time spent after

```
/* Wait for interrupt to wake us up */ mcr p15, 0, r0, c7, c0, 4
```

Yes, we need to reach the atmel_serial resume handler before the UART gets any chance to recognize the character. It's probably too late for the first one, but it may catch the next one assuming the baud rate isn't too high.

I believe, that we loose quite many characters. We are running at 32/16kHz (switching the master clock divider /2 off proved unstable) and this is not enough to even receive 9600 baud until we have the fast clock running again.

Some quick calculations:

For waking up, we need to switch on the main oscillator, wait until it stabilizes (default should be max: ~60ms), switch on plla, wait until it locks (2ms), switch on pll, wait until it locks (2ms), switch system to PLLA... And then set up the drivers again... On 115200 baud, this is almost 1kB!

So in many applications we could not use this. But this might still come handy in a lot of cases we can poll and find out what caused the data on the serial port etc. Or on applications, where this loss of data does not matter (like debug console where the resume is usable even if it does not wake up on the first byte).

--

Anti Sullin
Embedded Software Engineer
Artec Design LLC
Akadeemia tee 23A, 12618, Tallinn, Estonia
<http://www.artecdesign.ee>

--

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/>