

RE: [PATCH] SPI

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-09/8980.html>

dpervushin_at_gmail.com

Date: 09/30/05

To: "'David Brownell'" <david-b@pacbell.net>

Date: Fri, 30 Sep 2005 23:20:29 +0400

Hello all,

> *around the I/O model of a queue of async messages; and even
> names for some data structures.*

It seems we are talking about similar things, aren't we ?

> *<linux/spi/spi.h> ... main header*

> *<linux/spi/CHIP.h> ... platform_data, for CHIP.c driver*

>

> *Not all chips would need them, but it might be nice to have*

> *some place other than <linux/CHIP.h> for such things. The*

> *platform_data would have various important data that can't be*

> *... chip variants, initialization data, and similar stuff*

> *that differs between boards is knowable only by*

> *board-specific init code, yet is needed by board-agnostic driver code.*

I would prefer not to have subdirectory spi in include/linux. Take a look to pci, for example. I guess that chip data are spi-bus specific, and should not be exported to world.

> *that way internally. But other drivers shouldn't be forced*

> *to allocate kernel threads when they don't need them.*

Really :) ? I'd like to have the worker thread for bus (and all devices on the bus) instead of several workqueues (one per each device on bus, right ?)

> *Hmm, this seems to be missing a few important things ... from*

> *the last SPI patch I posted to this list (see the URL right above):*

>

> *struct bus_type spi_bus_type = {*

> *.name = "spi",*

> *.dev_attrs = spi_dev_attrs,*

> *.match = spi_match_device,*

> *.hotplug = spi_hotplug,*

> *.suspend = spi_suspend,*

> *.resume = spi_resume,*

> *};*

>

> *That supports new-school "modprobe \$MODALIAS" hotplugging and*

> *.../modalias style coldplugging, as well as passing PM calls*

> *down to the drivers. (Those last recently got some tweaking,*

> *to work better through sysfs.) And the core is STILL only*

> *about 2 KB on ARM; significantly less than yours.*

Are you counting bytes on your sources ? Or bytes in object files ? As for spi_bus_type, I agree. Hotplu/suspend/resume have to be included.

> *You don't seem to have any ability to record essential
> board-specific information that the drivers will need. I
> hope you're not planning on making that stuff clutter up the
> driver files?? board-specific.c files seem the better model,
> with a way to pass that data to the drivers that need it
> (using the driver model).*

>
> *That minimally includes stuff like the IRQ used by that chip,
> the clock rate it supports on this board, and the SPI
> clocking mode (0, 1, 2, 3) used to get data in and out of the
> chip. But there seem to be a few other things needed too,
> given the ways SPI chips tweak the protocol.*

This is responsibility of bus driver. The driver for device on the SPI bus might request the hardware info from the bus driver, which is referenced via spi_device->device->parent.

>
>
> > + /*
> > + * all messages for current
> > + * selected_device
> > + * are processed.
> > + * let's switch to another device
> > + */

>
> *Why are you hard-wiring such an unfair scheduling policy ...
> and preventing use of better ones? I'd use FIFO rather than
> something as unfair as that; and FIFO is much simpler to code, too.*

OK, the policy is hardcoded and seems to be not the only available. This can be solved by adding a function to pull out the message that is "next by current". Does this sound reasonable ?

>
>
> > +{
> > + int ret;
> > + struct spimsg *msg = spimsg_alloc(dev, SPI_M_RD, len, NULL);
> > +
> > + ret = spi_transfer(msg, NULL);
> > + memcpy(buf, spimsg_buffer_rd(msg), len);

>
> *I don't really understand why you'd want to make this so
> expensive though. Why not just do the IO directly into the
> buffer provided for that purpose? One controller might
> require dma bounce buffers; but don't penalize all others by
> imposing those same costs.*

Drivers might want to allocate their own buffers, for example, using dma_alloc_coherent. Such drivers also need to store the dma handle somewhere. Drivers might use pre-allocated buffers.

>

Linux-Kernel: RE: [PATCH] SPI

> Also, `spimsg_alloc()` is huge ... even if you expect the
> inliner will remove some of it. It's doing several dynamic
> allocations. I honestly don't understand why there's a need
> for even *_one_* dynamic allocation in this "core" code path
> (much less the memcpy).

The allocations might be avoided if drivers provide their callback to
"allocate" buffer. Then, there is the only alloc -- for `spi_msg` itself

> Also, you don't have any "board specific init" component in
> this code...

`spi_bus_populate` calls the callback to initialize device with `void*` context.

```
>
>
>> + +-----+ +-----+
>> + | platform_bus || spi_bus |
>> + +-----+ +-----+
>> + |..| |
>> + |..|-----+ +-----+
>> + +-----+ | is parent to | SPI devices |
>> + | SPI busses | +-----> | |
>> + +-----+ +-----+
>> + | |
>> + +-----+ +-----+
>> + | SPI bus driver || SPI device driver |
>> + +-----+ +-----+
>
```

> That seems wierd even if I assume "platform_bus" is just an example.
> For example there are two rather different "spi bus" notions
> there, and it looks like neither one is the physical parent
> of any SPI device ...

"SPI busses" means several 'struct device' that corresponds to real device
that acts as spi controller. "spi_bus" is the variable of type "bus_type"

```
>> + msg->devbuf_rd = drv->alloc ?
>> + drv->alloc(len, GFP_KERNEL) : kcalloc(len,
> GFP_KERNEL);
>> + msg->databuf_rd = drv->get_buffer ?
>> + drv->get_buffer(device, msg->devbuf_rd) :
> msg->devbuf_rd;
```

> Oy. More dynamic allocation. (Repeated for write buffers
> too ...) See above; don't force such costs on all drivers,
> few will ever need it.

That's not necessarily allocation. That depends on driver that uses
`spimsg_alloc`, and possibly provides callback for allocating
buffers/accessing them

```
>> + #define SPI_MAJOR 153
```

```
--
cheers, dmitry pervushin
```

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in

Linux-Kernel: RE: [PATCH] SPI

the body of a message to majordomo@vger.kernel.org

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

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