

Re: [PATCH] uio: User IRQ Mode

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-07/msg01207.html>

- *From:* "Magnus Damm" <magnus.damm@xxxxxxxxxx>
 - *Date:* Thu, 3 Jul 2008 19:23:34 +0900
-

Hi again Alan,

On Wed, Jul 2, 2008 at 8:11 PM, Alan Cox <alan@xxxxxxxxxxxxxxxxxxxxxx> wrote:

On Wed, 02 Jul 2008 19:59:51 +0900
Magnus Damm <magnus.damm@xxxxxxxxxx> wrote:

From: Uwe Kleine-König <Uwe.Kleine-Koenig@xxxxxxxxxx>

This patch adds a "User IRQ Mode" to UIO. In this mode the user space driver is responsible for acknowledging and re-enabling the interrupt. Shared interrupts are not supported by this mode.

This doesn't work even for some non shared interrupts.

If I take a level triggered interrupt then the IRQ handler code must clear the IRQ before the line can be unmasked.

It might work for edge triggered providing you don't get too many edges before you respond (in which case we will decide its a stuck IRQ and turn it off for good).

Below is a little description of how this is supposed to work, see point 5, 8, 9.

--- Regular in-kernel driver approach ---

1. hardware device signals the processor
-> interrupt signal to interrupt controller gets asserted
2. interrupt controller passes through interrupt signal
the source is enabled so the interrupt controller does not mask
-> interrupt signal to processor gets asserted
3. processor saves context and starts executing kernel interrupt code

Re: [PATCH] uio: User IRQ Mode

the processor will mask interrupts for this certain interrupt level

4. mask_ack_irq()

this modifies the state of the interrupt controller

→ interrupt signal to processor is no longer asserted

5. kernel interrupt handler acknowledges interrupt

this modifies the state of the hardware device

→ interrupt signal to interrupt controller is no longer asserted

6. unmask_irq() – unmask interrupt controller

this modifies the state of the interrupt controller

→ interrupt signal to processor is let through, but is no longer asserted

7. processor restores context

this will unmask interrupts for the current interrupt level

--- UIO "User IRQ Mode" approach ---

1. hardware device signals the processor

→ interrupt signal to interrupt controller gets asserted

2. interrupt controller passes through interrupt signal

the source is enabled so the interrupt controller does not mask

→ interrupt signal to processor gets asserted

3. processor saves context and starts executing kernel interrupt code

the processor will mask interrupts for this certain interrupt level

4. mask_ack_irq()

this modifies the state of the interrupt controller

→ interrupt signal to processor is no longer asserted

5. kernel interrupt handler _disables_ interrupts, notifies user space

this may modify the state of the interrupt controller

→ interrupt signal from device to interrupt controller is still asserted

→ interrupt source is still masked by interrupt controller

6. unmask_irq() – unmask interrupt controller

this may modify the state of the interrupt controller

→ interrupt source is still masked by interrupt controller

7. processor restores context

this will unmask interrupts for the current interrupt level

→ interrupt signal from device to interrupt controller is still asserted

→ interrupt source is still masked by interrupt controller

8. user space driver get scheduled, acknowledges interrupt

this modifies the state of the hardware device

→ interrupt signal from device to interrupt controller gets deasserted

→ interrupt source is still masked by interrupt controller

Re: [PATCH] uio: User IRQ Mode

Re: [PATCH] uio: User IRQ Mode

9. user space driver uses write() to re-enable the interrupt
this modifies the state of the interrupt controller
-> interrupt source is no longer masked by interrupt controller

Hope this clarifies a bit. Thank for your feedback!

/ magnus

--

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/>