

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

# RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

---

*Source:* <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-01/msg05575.html>

---

- *From:* Marc St-Jean <Marc\_St-Jean@xxxxxxxxxxxxxxxx>
  - *Date:* Mon, 22 Jan 2007 12:34:08 -0800
- 

CCing to linux-kernel as per AC's suggestion...

----- Original Message -----

Subject: RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git master  
Date: Mon, 22 Jan 2007 10:11:04 -0800  
From: Marc St-Jean  
To: Sergei Shtylyov  
CC: <linux-mips@xxxxxxxxxxxxxxxx>, <linux-serial@xxxxxxxxxxxxxxxx>

-----Original Message-----

From: Sergei Shtylyov [<mailto:sshtylyov@xxxxxxxxxxxxxxxx>]  
Sent: Friday, January 19, 2007 9:05 AM  
To: Marc St-Jean  
Cc: linux-mips@xxxxxxxxxxxxxxxx; linux-serial@xxxxxxxxxxxxxxxx  
Subject: Re: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git master

Hello.

Marc St-Jean wrote:

Here is a serial driver patch for the PMC-Sierra MSP71xx device.

There are three different fixes:

1. Fix for THREE errata
2. Fix for Busy Detect on LCR write
3. Workaround for interrupt/data concurrency issue

The first fix is handled cleanly using a UART\_BUG\_\* flag.

Hm, I wouldn't call it clean...

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

Relative to the other two. Any recommended improvements on this one?

The second and third fixes use platform tests. I couldn't think of a good way to implement them without using tests and not increase code and structure sizes for platforms not requiring them.

Does anyone have any suggestions on implementing these without the platform flag?

Thanks,  
Marc

Signed-off-by: Marc St-Jean <Marc\_St-Jean@xxxxxxxxxxxxxxxx>

Index: linux\_2\_6/drivers/serial/8250.c

=====  
RCS file: linux\_2\_6/drivers/serial/8250.c,v retrieving revision  
1.1.1.7 retrieving revision 1.9 diff -u -r1.1.1.7 -r1.9  
--- linux\_2\_6/drivers/serial/8250.c 19 Oct 2006 21:00:58

-0000 1.1.1.7

+++ linux\_2\_6/drivers/serial/8250.c 19 Oct 2006 22:08:15

-0000 1.9

```
@@ -44,6 +44,10 @@  
#include <asm/io.h>  
#include <asm/irq.h>  
  
+#ifdef CONFIG_PMC_MSP  
+#include <msp_regs.h>  
+#endif  
+
```

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

```
#include "8250.h"

/*
@@ -130,6 +134,9 @@
unsigned char mcr_mask; /* mask of user bits */
unsigned char mcr_force; /* mask of
forced bits */

unsigned char lsr_break_flag;
+#ifdef CONFIG_PMC_MSP
+ int dwapb_lcr;

/* saved LCR for DW APB WAR */

+#endif
```

There was already 'lcr' field there, couldn't it be used?

Possibly but the current driver doesn't always save the LCR value before trying to write it. For example see serial8250\_set\_termios()  
serial\_out(up, UART\_LCR, cval); /\* reset DLAB \*/  
up->lcr = cval; /\* Save LCR \*/

It's impossible to ensure that going forward the driver will always save the value before writing it. We need to have it saved before since the write will cause an interrupt, hence the save in serial\_out.

If that's not acceptable I can audit the driver and ensure all LCR writes are first saved. Maybe comments would reduce the chance of future updates breaking this?

```
@@ -333,6 +340,10 @@
static void
serial_out(struct uart_8250_port *up, int offset, int value)
{
+#ifdef CONFIG_PMC_MSP
+ /* Save the offset before it's remapped */
+ int dwapb_offset = offset;
+#endif
offset = map_8250_out_reg(up, offset) << up->port.regshift;

switch (up->port.iotype) {
@@ -342,7 +353,19 @@
break;

case UPIO_MEM:
```

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

```
+ #ifdef CONFIG_PMC_MSP
+ /* Save the LCR value so it can be re-written when a
+ * Busy Detect interrupt occurs. */
+ if (dwapb_offset == UART_LCR)
+ up->dwapb_lcr = value;
+ #endif
writeb(value, up->port.membase + offset);
+ #ifdef CONFIG_PMC_MSP
+ /* Re-read the IER to ensure any interrupt disabling has
+ * completed before proceeding with ISR. */
+ if (dwapb_offset == UART_IER)
+ value = serial_in(up, dwapb_offset); #endif
break;
```

Hm, was there really a need for #ifdef mess here?  
I'd vote for introducing new UPIO\_\* here, like was done  
for TSi10x UARTs just for the same reason.

I'm willing to do this, however IIRC there was a thread in late Sept.  
which came to the conclusion that UPIO were to be used for different  
access methods and not UART types. Do you see this as an exception?

In the second #ifdef CONFIG\_PMC\_MSP above the workaround is required  
because of SoC interrupt design which is out-of-band with respect to r/w  
of UART registers, it's not specific to the DesignWare UART.

```
@@ -1016,6 +1039,17 @@
up->bugs |= UART_BUG_NOMSR;
#endif

+ /* Workaround:
+ * The DesignWare SoC UART part has a bug for all
+ * versions before 3.03a (2005-07-18)
+ * In brief, this is a non-standard 16550 in that the
```

THRE interrupt

```
+ * will not re-assert itself simply by disabling and
re-enabling the
```

```
+ * THRI bit in the IER, it is only re-enabled if a
character is actually
```

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

```
+ * sent out.  
+ */  
+ if( up->port.flags & UPF_DW_THRE_BUG )  
+ up->bugs |= UART_BUG_DWTHRE;  
+  
serial_outp(up, UART_LCR, save_lcr);  
  
if (up->capabilities != uart_config[up->port.type].flags) { @@  
-1141,6 +1175,12 @@  
iir = serial_in(up, UART_IIR);  
if (lsr & UART_LSR_TEMT && iir &
```

UART\_IIR\_NO\_INT)

```
transmit_chars(up);  
+ } else if (up->bugs & UART_BUG_DWTHRE) {  
+ unsigned char lsr, iir;  
+ lsr = serial_in(up, UART_LSR);  
+ iir = serial_in(up, UART_IIR);  
+ if (lsr & UART_LSR_THRE)  
+ transmit_chars(up);
```

I don't see how this *\*really\** differs from the UART\_BUG\_TXEN case.  
Have you tried *\*that\** workaround?

I didn't write the code so I haven't tried it personally but I believe  
it was investigated.

It looks like the results of the bugs are similar but the causes are  
different. In the UART\_BUG\_TXEN case, if I understand the comment  
correctly, NO interrupt is generated on enabling THRI. This is verified  
by looking for Transmitter Empty (0x40) and no interrupt.

In the UART\_BUG\_DWTHRE case an interrupt IS generated on enabling THRI.  
However no new THRE interrupts will occur until a new character is  
written to the THR and it's sent. This is verified by looking for  
Transmit-Hold-Register Empty (0x20).

In any case, looks like this errata is auto-detectable just like

UART\_BUG\_TXEN.

I don't know how we would be able to test this without sending junk test  
characters to the console port. We currently enable the bug flag in our  
platform setup code when we know the bug is present from the SoC  
version. Is that not acceptable?

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

```
@@ -1366,6 +1406,31 @@
handled = 1;

end = NULL;
+#ifdef CONFIG_PMC_MSP
+ } else if ((iir & UART_IER_BUSY) == UART_IER_BUSY) {
```

Hm, masking IIR with IER mask, is this correct? Doubt it.

I agree, that was badly named. I've changed it to UART\_IIR\_BUSY for the next spin.

```
+ /*
+ * The MSP (DesignWare APB UART) serial
subsystem has a
+ * non-standard interrupt condition
(0x7) which means
+ * that the LCR was written while the
UART was busy, so
+ * the LCR was not actually written.
It is cleared by
+ * reading the special non-standard
extended UART status
+ * register.
+ */
+ unsigned int tmp;
+ if( up->port.line == 0 )
+ tmp = *UART0_STATUS_REG;
+ else
+ tmp = *UART1_STATUS_REG;
+
+ /* Check if saved on LCR write */
+ if( up->dwapb_lcr != -1 )
+ serial_outp(up, UART_LCR,
```

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

```
up->dwapb_lcr);
```

```
+ else  
+ printk(KERN_ERR "serial8250:
```

```
UART BUSY, no LCR write!\n" );
```

```
+  
+ handled = 1;  
+ end = NULL;  
+ #endif
```

Not sure if this also shouldn't be handled in other places which check for interrupt status, like serial8250\_timeout()...

I can move the code to a function but I still see two issues:

A) We could move the code to a new function. We could also check for a new UART\_BUG\_\* flag before calling the function, but in reality this is a feature of the UART not a bug.

B) How to eliminate the platform #ifdef. How to pass in the address of the UARTx\_STATUS\_REG in a platform independent way?

We could add it to one of the data structures like uart\_port, but it would need an #ifdef in the header file.

[...]

```
Index: linux_2_6/drivers/serial/8250.h
```

```
=====  
RCS file: linux_2_6/drivers/serial/8250.h,v retrieving revision  
1.1.1.6 retrieving revision 1.4 diff -u -r1.1.1.6 -r1.4  
--- linux_2_6/drivers/serial/8250.h 19 Oct 2006 21:00:58
```

```
-0000 1.1.1.6
```

```
+++ linux_2_6/drivers/serial/8250.h 19 Oct 2006 22:08:15
```

```
-0000 1.4
```

```
@@ -49,6 +49,7 @@  
#define UART_BUG_QUOT (1 << 0) /* UART has
```

```
buggy quot LSB */
```

```
#define UART_BUG_TXEN (1 << 1) /* UART has
```

```
buggy TX IIR status */
```

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

```
#define UART_BUG_NOMSR (1 << 2) /* UART has
buggy MSR status bits (Au1x00) */
+#define UART_BUG_DWTHRE (1 << 3) /* UART has buggy
DesignWare THRE interrupt re-assertion */
```

```
#define PROBE_RSA (1 << 0)
#define PROBE_ANY (~0)
Index: linux_2_6/include/linux/serial_core.h
=====
RCS file: linux_2_6/include/linux/serial_core.h,v
retrieving revision 1.1.1.7
retrieving revision 1.5
diff -u -r1.1.1.7 -r1.5
--- linux_2_6/include/linux/serial_core.h 19 Oct 2006
```

21:01:02 -0000 1.1.1.7

```
+++ linux_2_6/include/linux/serial_core.h 19 Oct 2006
```

22:08:16 -0000 1.5

```
@@ -258,6 +258,8 @@
#define UPF_CONS_FLOW ((__force upf_t) (1 << 23))
#define UPF_SHARE_IRQ ((__force upf_t) (1 << 24))
#define UPF_BOOT_AUTOCONF ((__force upf_t) (1 << 28))
+#define UPF_DW_THRE_BUG ((__force upf_t)(1 <<
```

29)) /\* DesignWare THRE hardware BUG

The need for the new flag seems doubtful to me.

Please see my earlier comment on this issue being different than  
UART\_BUG\_TXEN.

+

```
* (cannot be
autodetected) */
```

The patch is linewrapped

```
Index: linux_2_6/include/linux/serial_reg.h
=====
```

RE: [PATCH] serial driver PMC MSP71xx, kernel linux-mips.git mast er

```
RCS file: linux_2_6/include/linux/serial_reg.h,v
retrieving revision 1.1.1.2
retrieving revision 1.3
diff -u -r1.1.1.2 -r1.3
--- linux_2_6/include/linux/serial_reg.h 19 Oct 2006
```

18:29:50 -0000 1.1.1.2

```
+++ linux_2_6/include/linux/serial_reg.h 19 Oct 2006
```

19:45:04 -0000 1.3

```
@@ -218,6 +218,10 @@
#define UART_FCR_PXAR16 0x80 /* receive FIFO threshold = 16 */
#define UART_FCR_PXAR32 0xc0 /* receive FIFO threshold = 32 */

+/*
+ * DesignWare APB UART
+ */
+#define UART_IER_BUSY 0x07 /* Busy Detect */
```

Are you sure it's not \*IIR\* value? Doesn't look like interrupt mask for IER. And IIR value of 7 already means something else, namely, no interrupt and receiver status. Hm...

As mentioned earlier I have now renamed this UART\_IIR\_BUSY.

From serial\_reg.h, the receiver status is 0x06 and no interrupt is 0x01. I thought these couldn't both be set at the same time? In any case this is how DesignWare implemented the status so we can't change it now.

Thanks for the feedback,  
Marc

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxxx  
More majordomo info at <http://vger.kernel.org/majordomo-info.html>  
Please read the FAQ at <http://www.tux.org/lkml/>