

[PATCH] 8250 UART backup timer

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2006-06/msg09187.html>

- *From:* Alex Williamson <alex.williamson@xxxxxx>
 - *Date:* Tue, 27 Jun 2006 13:04:14 -0600
-

The patch below works around a minor bug found in the UART of the remote management card used in many HP ia64 and parisc servers (aka the Diva UARTs). The problem is that the UART does not reassert the THRE interrupt if it has been previously cleared and the IIR THRI bit is re-enabled. This can produce a very annoying failure mode when used as a serial console, allowing a boot/reboot to hang indefinitely until an RX interrupt kicks it into working again (ie. an unattended reboot could stall).

To solve this problem, a backup timer is introduced that runs alongside the standard interrupt driven mechanism. This timer wakes up periodically, checks for a hang condition and gets characters moving again. This backup mechanism is only enabled if the UART is detected as having this problem, so systems without these UARTs will have no additional overhead.

This version of the patch incorporates previous comments from Pavel and removes races in the bug detection code. The test is now done before the irq linking to prevent races with interrupt handler clearing the THRE interrupt. Short delays and syncs are also added to ensure the device is able to update register state before the result is tested. Comments? Thanks,

Alex

Signed-off-by: Alex Williamson <alex.williamson@xxxxxx>

```
----  
diff -r e3554576c29a drivers/serial/8250.c  
--- a/drivers/serial/8250.c Sun Jun 18 02:00:07 2006 +0000  
+++ b/drivers/serial/8250.c Tue Jun 27 12:27:03 2006 -0600  
@@ -353,6 +353,23 @@ serial_out(struct uart_8250_port *up, in  
 }  
 }  
  
+static void  
+serial_out_sync(struct uart_8250_port *up, int offset, int value)  
+{
```

[PATCH] 8250 UART backup timer

```
+ switch (up->port.iotype) {
+ case UPIO_MEM:
+ case UPIO_MEM32:
+#ifdef CONFIG_SERIAL_8250_AU1X00
+ case UPIO_AU:
+#endif
+ serial_out(up, offset, value);
+ (void)serial_in(up, UART_LCR); /* safe, no side-effects */
+ break;
+ default:
+ serial_out(up, offset, value);
+ }
+}
+
+/*
+ * We used to support using pause I/O for certain machines. We
+ * haven't supported this for a while, but just in case it's badly
+ @@ -1436,6 +1453,13 @@ static void serial_unlink_irq_chain(stru
serial_do_unlink(i, up);
}

+/* Base timer interval for polling */
+static inline int
+poll_timeout(int timeout)
+{
+ return timeout > 6 ? (timeout / 2 - 2) : 1;
+}
+
+/*
+ * This function is used to handle ports that do not have an
+ * interrupt. This doesn't work very well for 16450's, but gives
+ @@ -1445,16 +1469,52 @@ static void serial8250_timeout(unsigned
static void serial8250_timeout(unsigned long data)
{
struct uart_8250_port *up = (struct uart_8250_port *)data;
- unsigned int timeout;
unsigned int iir;

iir = serial_in(up, UART_IIR);
if (!(iir & UART_IIR_NO_INT))
serial8250_handle_port(up, NULL);

- timeout = up->port.timeout;
- timeout = timeout > 6 ? (timeout / 2 - 2) : 1;
- mod_timer(&up->timer, jiffies + timeout);
+ mod_timer(&up->timer, jiffies + poll_timeout(up->port.timeout));
+}
+
+static void serial8250_backup_timeout(unsigned long data)
+{
+ struct uart_8250_port *up = (struct uart_8250_port *)data;
```


[PATCH] 8250 UART backup timer

```
+
+ /* Wait up to 10ms for the character(s) to be sent. */
+ do {
+ status = serial_in(up, UART_LSR);
+
+ if (status & UART_LSR_BI)
+ up->lcr_break_flag = UART_LSR_BI;
+
+ if (--timeout == 0)
+ break;
+ udelay(1);
+ } while ((status & bits) != bits);
+
+ /* Wait up to 1s for flow control if necessary */
+ if (up->port.flags & UPF_CONS_FLOW) {
+ timeout = 1000000;
+ while (--timeout &&
+ ((serial_in(up, UART_MSR) & UART_MSR_CTS) == 0))
+ udelay(1);
+ }
+ }

static int serial8250_startup(struct uart_port *port)
@@ -1598,18 +1688,50 @@ static int serial8250_startup(struct uar
serial_outp(up, UART_LCR, 0);
}

+ if (is_real_interrupt(up->port.irq)) {
+ /*
+ * Test for UARTs that do not reassert THRE when the
+ * transmitter is idle and the interrupt has already
+ * been cleared. Real 16550s should always reassert
+ * this interrupt whenever the transmitter is idle and
+ * the interrupt is enabled. Delays are necessary to
+ * allow register changes to become visible.
+ */
+ spin_lock_irqsave(&up->port.lock, flags);
+
+ wait_for_xmitr(up, UART_LSR_THRE);
+ serial_out_sync(up, UART_IER, UART_IER_THRI);
+ udelay(1); /* allow THRE to set */
+ (void)serial_in(up, UART_IIR);
+ serial_out(up, UART_IER, 0);
+ serial_out_sync(up, UART_IER, UART_IER_THRI);
+ udelay(1); /* allow a working UART time to re-assert THRE */
+ iir = serial_in(up, UART_IIR);
+ serial_out(up, UART_IER, 0);
+
+ spin_unlock_irqrestore(&up->port.lock, flags);
+
+ /*
```

[PATCH] 8250 UART backup timer

```
+ * If the interrupt is not reasserted, setup a timer to
+ * kick the UART on a regular basis.
+ */
+ if (iir & UART_IIR_NO_INT) {
+ pr_debug("ttyS%d - using backup timer\n", port->line);
+ up->timer.function = serial8250_backup_timeout;
+ up->timer.data = (unsigned long)up;
+ mod_timer(&up->timer, jiffies +
+ poll_timeout(up->port.timeout) + HZ/5);
+ }
+ }
+
+/*
+ * If the "interrupt" for this port doesn't correspond with any
+ * hardware interrupt, we use a timer-based system. The original
+ * driver used to do this with IRQ0.
+ */
+ if (!is_real_interrupt(up->port.irq)) {
+ unsigned int timeout = up->port.timeout;
+
+ timeout = timeout > 6 ? (timeout / 2 - 2) : 1;
+
+ up->timer.data = (unsigned long)up;
+ mod_timer(&up->timer, jiffies + timeout);
+ mod_timer(&up->timer, jiffies + poll_timeout(up->port.timeout));
+ } else {
+ retval = serial_link_irq_chain(up);
+ if (retval)
+ @@ -1725,9 +1847,9 @@ static void serial8250_shutdown(struct u
+ */
+ (void) serial_in(up, UART_RX);
+
+ if (!is_real_interrupt(up->port.irq))
+ del_timer_sync(&up->timer);
+ else
+ del_timer_sync(&up->timer);
+ up->timer.function = serial8250_timeout;
+ if (is_real_interrupt(up->port.irq))
+ serial_unlink_irq_chain(up);
+ }
+
+ @@ -2187,36 +2309,6 @@ serial8250_register_ports(struct uart_dr
+
+ #ifdef CONFIG_SERIAL_8250_CONSOLE
+
+ #define BOTH_EMPTY (UART_LSR_TEMT | UART_LSR_THRE)
+
+ /*
+ * Wait for transmitter & holding register to empty
+ */
+ static inline void wait_for_xmitr(struct uart_8250_port *up, int bits)
```

[PATCH] 8250 UART backup timer

[PATCH] 8250 UART backup timer

```
-{
- unsigned int status, tmout = 10000;
-
- /* Wait up to 10ms for the character(s) to be sent. */
- do {
- status = serial_in(up, UART_LSR);
-
- if (status & UART_LSR_BI)
- up->lsr_break_flag = UART_LSR_BI;
-
- if (--tmout == 0)
- break;
- udelay(1);
- } while ((status & bits) != bits);
-
- /* Wait up to 1s for flow control if necessary */
- if (up->port.flags & UPF_CONS_FLOW) {
- tmout = 1000000;
- while (--tmout &&
- ((serial_in(up, UART_MSR) & UART_MSR_CTS) == 0))
- udelay(1);
- }
- }
-
static void serial8250_console_putchar(struct uart_port *port, int ch)
{
struct uart_8250_port *up = (struct uart_8250_port *)port;
```

-

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/>