

[PATCH] CRISv10 serial driver rewrite

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-11/msg00534.html>

- *From:* Jesper Nilsson <jesper.nilsson@xxxxxxxx>
 - *Date:* Fri, 2 Nov 2007 10:34:32 +0100
-

Resubmission of the new and improved serial driver for CRISv10, this time with both patches in the same mail and with hopefully useful subject.

- Removed CVS tags.
- Removed defines and comments for CRIS_BUF_SIZE and TTY_THRESHOLD_THROTTLE (no longer used).
- Merge of CRISv10 from Axis internal CVS.

Signed-off-by: Jesper Nilsson <jesper.nilsson@xxxxxxxx>

drivers/serial/crisv10.c | 1298 ++++++-----
drivers/serial/crisv10.h | 151 +++
2 files changed, 594 insertions(+), 855 deletions(-)

```
diff --git a/drivers/serial/crisv10.c b/drivers/serial/crisv10.c
index f523cdf..965ed6a 100644
--- a/drivers/serial/crisv10.c
+++ b/drivers/serial/crisv10.c
@@ -1,426 +1,10 @@
-/* $Id: serial.c,v 1.25 2004/09/29 10:33:49 starvik Exp $
- *
+/*
+ * Serial port driver for the ETRAX 100LX chip
+ *
- * Copyright (C) 1998, 1999, 2000, 2001, 2002, 2003 Axis Communications AB
+ * Copyright (C) 1998-2007 Axis Communications AB
+ *
+ * Many, many authors. Based once upon a time on serial.c for 16x50.
+ *
- * $Log: serial.c,v $
- * Revision 1.25 2004/09/29 10:33:49 starvik
- * Resolved a dealock when printing debug from kernel.
- *
- * Revision 1.24 2004/08/27 23:25:59 johana
- * rs_set_termios() must call change_speed() if c_iflag has changed or
- * automatic XOFF handling will be enabled and transmitter will stop
- * if 0x13 is received.
```

[PATCH] CRISv10 serial driver rewrite

- *
- * Revision 1.23 2004/08/24 06:57:13 starvik
- * More whitespace cleanup
- *
- * Revision 1.22 2004/08/24 06:12:20 starvik
- * Whitespace cleanup
- *
- * Revision 1.20 2004/05/24 12:00:20 starvik
- * Big merge of stuff from Linux 2.4 (e.g. manual mode for the serial port).
- *
- * Revision 1.19 2004/05/17 13:12:15 starvik
- * Kernel console hook
- * Big merge from Linux 2.4 still pending.
- *
- * Revision 1.18 2003/10/28 07:18:30 starvik
- * Compiles with debug info
- *
- * Revision 1.17 2003/07/04 08:27:37 starvik
- * Merge of Linux 2.5.74
- *
- * Revision 1.16 2003/06/13 10:05:19 johana
- * Help the user to avoid trouble by:
- * Forcing mixed mode for status/control lines if not all pins are used.
- *
- * Revision 1.15 2003/06/13 09:43:01 johana
- * Merged in the following changes from os/linux/arch/cris/drivers/serial.c
- * + some minor changes to reduce diff.
- *
- * Revision 1.49 2003/05/30 11:31:54 johana
- * Merged in change-branch--serial9bit that adds CMSPAR support for sticky
- * parity (mark/space)
- *
- * Revision 1.48 2003/05/30 11:03:57 johana
- * Implemented rs_send_xchar() by disabling the DMA and writing manually.
- * Added e100_disable_txdma_channel() and e100_enable_txdma_channel().
- * Fixed rs_throttle() and rs_unthrottle() to properly call rs_send_xchar
- * instead of setting info->x_char and check the CRTSCTS flag before
- * controlling the rts pin.
- *
- * Revision 1.14 2003/04/09 08:12:44 pkj
- * Corrected typo changes made upstream.
- *
- * Revision 1.13 2003/04/09 05:20:47 starvik
- * Merge of Linux 2.5.67
- *
- * Revision 1.11 2003/01/22 06:48:37 starvik
- * Fixed warnings issued by GCC 3.2.1
- *
- * Revision 1.9 2002/12/13 09:07:47 starvik
- * Alert user that RX_TIMEOUT_TICKS==0 doesn't work
- *

[PATCH] CRISv10 serial driver rewrite

- * Revision 1.8 2002/12/11 13:13:57 starvik
- * Added arch/ to v10 specific includes
- * Added fix from Linux 2.4 in serial.c (flush_to_flip_buffer)
- *
- * Revision 1.7 2002/12/06 07:13:57 starvik
- * Corrected work queue stuff
- * Removed CONFIG_ETRAX_SERIAL_FLUSH_DMA_FAST
- *
- * Revision 1.6 2002/11/21 07:17:46 starvik
- * Change static inline to extern inline where otherwise outlined with gcc-3.2
- *
- * Revision 1.5 2002/11/14 15:59:49 starvik
- * Linux 2.5 port of the latest serial driver from 2.4. The work queue stuff
- * probably doesn't work yet.
- *
- * Revision 1.42 2002/11/05 09:08:47 johana
- * Better implementation of rs_stop() and rs_start() that uses the XOFF
- * register to start/stop transmission.
- * change_speed() also initalises XOFF register correctly so that
- * auto_xoff is enabled when IXON flag is set by user.
- * This gives fast XOFF response times.
- *
- * Revision 1.41 2002/11/04 18:40:57 johana
- * Implemented rs_stop() and rs_start().
- * Simple tests using hwtestserial indicates that this should be enough
- * to make it work.
- *
- * Revision 1.40 2002/10/14 05:33:18 starvik
- * RS-485 uses fast timers even if SERIAL_FAST_TIMER is disabled
- *
- * Revision 1.39 2002/09/30 21:00:57 johana
- * Support for CONFIG_ETRAX_SERx_DTR_RI_DSR_CD_MIXED where the status and
- * control pins can be mixed between PA and PB.
- * If no serial port uses MIXED old solution is used
- * (saves a few bytes and cycles).
- * control_pins struct uses masks instead of bit numbers.
- * Corrected dummy values and polarity in line_info() so
- * /proc/tty/driver/serial is now correct.
- * (the E100_xxx_GET() macros is really active low – perhaps not obvious)
- *
- * Revision 1.38 2002/08/23 11:01:36 starvik
- * Check that serial port is enabled in all interrupt handlers to avoid
- * restarts of DMA channels not assigned to serial ports
- *
- * Revision 1.37 2002/08/13 13:02:37 bjornw
- * Removed some warnings because of unused code
- *
- * Revision 1.36 2002/08/08 12:50:01 starvik
- * Serial interrupt is shared with synchronous serial port driver
- *
- * Revision 1.35 2002/06/03 10:40:49 starvik

[PATCH] CRISv10 serial driver rewrite

- * Increased RS–485 RTS toggle timer to 2 characters
- *
- * Revision 1.34 2002/05/28 18:59:36 johana
- * Whitespace and comment fixing to be more like etrax100ser.c 1.71.
- *
- * Revision 1.33 2002/05/28 17:55:43 johana
- * RS–485 uses FAST_TIMER if enabled, and starts a short (one char time)
- * timer from transmit_chars (interrupt context).
- * The timer toggles RTS in interrupt context when expired giving minimum
- * latencies.
- *
- * Revision 1.32 2002/05/22 13:58:00 johana
- * Renamed rs_write() to raw_write() and made it inline.
- * New rs_write() handles RS–485 if configured and enabled
- * (moved code from e100_write_rs485()).
- * RS–485 ioctl's uses copy_from_user() instead of verify_area().
- *
- * Revision 1.31 2002/04/22 11:20:03 johana
- * Updated copyright years.
- *
- * Revision 1.30 2002/04/22 09:39:12 johana
- * RS–485 support compiles.
- *
- * Revision 1.29 2002/01/14 16:10:01 pkj
- * Allocate the receive buffers dynamically. The static 4kB buffer was
- * too small for the peaks. This means that we can get rid of the extra
- * buffer and the copying to it. It also means we require less memory
- * under normal operations, but can use more when needed (there is a
- * cap at 64kB for safety reasons). If there is no memory available
- * we panic(), and die a horrible death...
- *
- * Revision 1.28 2001/12/18 15:04:53 johana
- * Cleaned up write_rs485() – now it works correctly without padding extra
- * char.
- * Added sane default initialisation of rs485.
- * Added #ifdef around dummy variables.
- *
- * Revision 1.27 2001/11/29 17:00:41 pkj
- * 2kB seems to be too small a buffer when using 921600 bps,
- * so increase it to 4kB (this was already done for the elinux
- * version of the serial driver).
- *
- * Revision 1.26 2001/11/19 14:20:41 pkj
- * Minor changes to comments and unused code.
- *
- * Revision 1.25 2001/11/12 20:03:43 pkj
- * Fixed compiler warnings.
- *
- * Revision 1.24 2001/11/12 15:10:05 pkj
- * Total redesign of the receiving part of the serial driver.
- * Uses eight chained descriptors to write to a 4kB buffer.

[PATCH] CRISv10 serial driver rewrite

- * This data is then serialised into a 2kB buffer. From there it
- * is copied into the TTY's flip buffers when they become available.
- * A lot of copying, and the sizes of the buffers might need to be
- * tweaked, but all in all it should work better than the previous
- * version, without the need to modify the TTY code in any way.
- * Also note that erroneous bytes are now correctly marked in the
- * flag buffers (instead of always marking the first byte).
- *
- * Revision 1.23 2001/10/30 17:53:26 pkj
- * * Set info->uses_dma to 0 when a port is closed.
- * * Mark the timer1 interrupt as a fast one (SA_INTERRUPT).
- * * Call start_flush_timer() in start_receive() if
- * CONFIG_ETRAX_SERIAL_FLUSH_DMA_FAST is defined.
- *
- * Revision 1.22 2001/10/30 17:44:03 pkj
- * Use %lu for received and transmitted counters in line_info().
- *
- * Revision 1.21 2001/10/30 17:40:34 pkj
- * Clean-up. The only change to functionality is that
- * CONFIG_ETRAX_SERIAL_RX_TIMEOUT_TICKS(=5) is used instead of
- * MAX_FLUSH_TIME(=8).
- *
- * Revision 1.20 2001/10/30 15:24:49 johana
- * Added char_time stuff from 2.0 driver.
- *
- * Revision 1.19 2001/10/30 15:23:03 johana
- * Merged with 1.13.2 branch + fixed indentation
- * and changed CONFIG_ETRAX100_XYS to CONFIG_ETRAX_XYZ
- *
- * Revision 1.18 2001/09/24 09:27:22 pkj
- * Completed ext_baud_table[] in cflag_to_baud() and cflag_to_etrax_baud().
- *
- * Revision 1.17 2001/08/24 11:32:49 ronny
- * More fixes for the CONFIG_ETRAX_SERIAL_PORT0 define.
- *
- * Revision 1.16 2001/08/24 07:56:22 ronny
- * Added config ifdefs around ser0 irq requests.
- *
- * Revision 1.15 2001/08/16 09:10:31 bjarne
- * serial.c – corrected the initialization of rs_table, the wrong defines
- * where used.
- * Corrected a test in timed_flush_handler.
- * Changed configured to enabled.
- * serial.h – Changed configured to enabled.
- *
- * Revision 1.14 2001/08/15 07:31:23 bjarne
- * Introduced two new members to the e100_serial struct.
- * configured – Will be set to 1 if the port has been configured in .config
- * uses_dma – Should be set to 1 if the port uses DMA. Currently it is set
- * to 1
- * when a port is opened. This is used to limit the DMA interrupt

[PATCH] CRISv10 serial driver rewrite

- * routines to only manipulate DMA channels actually used by the
- * serial driver.
- *
- * Revision 1.13.2.2 2001/10/17 13:57:13 starvik
- * Receiver was broken by the break fixes
- *
- * Revision 1.13.2.1 2001/07/20 13:57:39 ronny
- * Merge with new stuff from etrax100ser.c. Works but haven't checked stuff
- * like break handling.
- *
- * Revision 1.13 2001/05/09 12:40:31 johana
- * Use DMA_NBR and IRQ_NBR defines from dma.h and irq.h
- *
- * Revision 1.12 2001/04/19 12:23:07 bjornw
- * CONFIG_RS485 -> CONFIG_ETRAX_RS485
- *
- * Revision 1.11 2001/04/05 14:29:48 markusl
- * Updated according to review remarks i.e.
- * –Use correct types in port structure to avoid compiler warnings
- * –Try to use IO_* macros whenever possible
- * –Open should never return –EBUSY
- *
- * Revision 1.10 2001/03/05 13:14:07 bjornw
- * Another spelling fix
- *
- * Revision 1.9 2001/02/23 13:46:38 bjornw
- * Spelling check
- *
- * Revision 1.8 2001/01/23 14:56:35 markusl
- * Made use of ser1 optional
- * Needed by USB
- *
- * Revision 1.7 2001/01/19 16:14:48 perf
- * Added kernel options for serial ports 234.
- * Changed option names from CONFIG_ETRAX100_XYZ to CONFIG_ETRAX_XYZ.
- *
- * Revision 1.6 2000/11/22 16:36:09 bjornw
- * Please marketing by using the correct case when spelling Etrax.
- *
- * Revision 1.5 2000/11/21 16:43:37 bjornw
- * Fixed so it compiles under CONFIG_SVINTO_SIM
- *
- * Revision 1.4 2000/11/15 17:34:12 bjornw
- * Added a timeout timer for flushing input channels. The interrupt-based
- * fast flush system should be easy to merge with this later (works the same
- * way, only with an irq instead of a system timer_list)
- *
- * Revision 1.3 2000/11/13 17:19:57 bjornw
- * * Incredibly, this almost complete rewrite of serial.c worked (at least
- * for output) the first time.
- *

[PATCH] CRISv10 serial driver rewrite

- * Items worth noticing:
- *
- * No Etrax100 port 1 workarounds (does only compile on 2.4 anyway now)
- * RS485 is not ported (why can't it be done in userspace as on x86 ?)
- * Statistics done through `async_ccount` – if any more stats are needed,
- * that's the place to put them or in an arch-dep version of it.
- * `timeout_interrupt` and the other fast timeout stuff not ported yet
- * There be dragons in this 3k+ line driver
- *
- * Revision 1.2 2000/11/10 16:50:28 bjornw
- * First shot at a 2.4 port, does not compile totally yet
- *
- * Revision 1.1 2000/11/10 16:47:32 bjornw
- * Added verbatim copy of rev 1.49 `etrax100ser.c` from `elinux`
- *
- * Revision 1.49 2000/10/30 15:47:14 tobiasa
- * Changed version number.
- *
- * Revision 1.48 2000/10/25 11:02:43 johana
- * Changed `%ul` to `%lu` in `printf`'s
- *
- * Revision 1.47 2000/10/18 15:06:53 pkj
- * Compile correctly with `CONFIG_ETRAX_SERIAL_FLUSH_DMA_FAST` and
- * `CONFIG_ETRAX_SERIAL_PROC_ENTRY` together.
- * Some clean-up of the `/proc/serial` file.
- *
- * Revision 1.46 2000/10/16 12:59:40 johana
- * Added `CONFIG_ETRAX_SERIAL_PROC_ENTRY` for statistics and debug info.
- *
- * Revision 1.45 2000/10/13 17:10:59 pkj
- * Do not flush DMAs while flipping TTY buffers.
- *
- * Revision 1.44 2000/10/13 16:34:29 pkj
- * Added a delay in `ser_interrupt()` for 2.3ms when an error is detected.
- * We do not know why this delay is required yet, but without it the
- * `irmaflash` program does not work (this was the program that needed
- * the `ser_interrupt()` to be needed in the first place). This should not
- * affect normal use of the serial ports.
- *
- * Revision 1.43 2000/10/13 16:30:44 pkj
- * New version of the fast flush of serial buffers code. This time
- * it is localized to the serial driver and uses a fast timer to
- * do the work.
- *
- * Revision 1.42 2000/10/13 14:54:26 bennyo
- * Fix for switching RTS when using rs485
- *
- * Revision 1.41 2000/10/12 11:43:44 pkj
- * Cleaned up a number of comments.
- *
- * Revision 1.40 2000/10/10 11:58:39 johana

[PATCH] CRISv10 serial driver rewrite

- * Made RS485 support generic for all ports.
- * Toggle rts in interrupt if no delay wanted.
- * WARNING: No true transmitter empty check??
- * Set d_wait bit when sending data so interrupt is delayed until
- * fifo flushed. (Fix tcdrain() problem)
- *
- * Revision 1.39 2000/10/04 16:08:02 bjornw
- * * Use virt_to_phys etc. for DMA addresses
- * * Removed CONFIG_FLUSH_DMA_FAST hacks
- * * Indentation fix
- *
- * Revision 1.38 2000/10/02 12:27:10 mattias
- * * added variable used when using fast flush on serial dma.
- * (CONFIG_FLUSH_DMA_FAST)
- *
- * Revision 1.37 2000/09/27 09:44:24 pkj
- * Uncomment definition of SERIAL_HANDLE_EARLY_ERRORS.
- *
- * Revision 1.36 2000/09/20 13:12:52 johana
- * Support for CONFIG_ETRAX_SERIAL_RX_TIMEOUT_TICKS:
- * Number of timer ticks between flush of receive fifo (1 tick = 10ms).
- * Try 0-3 for low latency applications. Approx 5 for high load
- * applications (e.g. PPP). Maybe this should be more adaptive some day...
- *
- * Revision 1.35 2000/09/20 10:36:08 johana
- * Typo in get_lsr_info()
- *
- * Revision 1.34 2000/09/20 10:29:59 johana
- * Let rs_chars_in_buffer() check fifo content as well.
- * get_lsr_info() might work now (not tested).
- * Easier to change the port to debug.
- *
- * Revision 1.33 2000/09/13 07:52:11 torbjore
- * Support RS485
- *
- * Revision 1.32 2000/08/31 14:45:37 bjornw
- * After sending a break we need to reset the transmit DMA channel
- *
- * Revision 1.31 2000/06/21 12:13:29 johana
- * Fixed wait for all chars sent when closing port.
- * (Used to always take 1 second!)
- * Added shadows for directions of status/ctrl signals.
- *
- * Revision 1.30 2000/05/29 16:27:55 bjornw
- * Simulator ifdef moved a bit
- *
- * Revision 1.29 2000/05/09 09:40:30 mattias
- * * Added description of dma registers used in timeout_interrupt
- * * Removed old code
- *
- * Revision 1.28 2000/05/08 16:38:58 mattias

[PATCH] CRISv10 serial driver rewrite

```
- * * Bugfix for flushing fifo in timeout_interrupt
- * Problem occurs when bluetooth stack waits for a small number of bytes
- * containing an event acknowledging free buffers in bluetooth HW
- * As before, data was stuck in fifo until more data came on uart and
- * flushed it up to the stack.
- *
- * Revision 1.27 2000/05/02 09:52:28 jonasd
- * Added fix for peculiar etrax behaviour when eop is forced on an empty
- * fifo. This is used when flashing the IRMA chip. Disabled by default.
- *
- * Revision 1.26 2000/03/29 15:32:02 bjornw
- * 2.0.34 updates
- *
- * Revision 1.25 2000/02/16 16:59:36 bjornw
- * * Receive DMA directly into the flip-buffer, eliminating an intermediary
- * receive buffer and a memcpy. Will avoid some overruns.
- * * Error message on debug port if an overrun or flip buffer overrun occurs.
- * * Just use the first byte in the flag flip buffer for errors.
- * * Check for timeout on the serial ports only each 5/100 s, not 1/100.
- *
- * Revision 1.24 2000/02/09 18:02:28 bjornw
- * * Clear serial errors (overrun, framing, parity) correctly. Before, the
- * receiver would get stuck if an error occurred and we did not restart
- * the input DMA.
- * * Cosmetics (indentation, some code made into inlines)
- * * Some more debug options
- * * Actually shut down the serial port (DMA irq, DMA reset, receiver stop)
- * when the last open is closed. Corresponding fixes in startup().
- * * rs_close() "tx FIFO wait" code moved into right place, bug & -> && fixed
- * and make a special case out of port 1 (R_DMA_CHx_STATUS is broken for that)
- * * e100_disable_rx/enable_rx just disables/enables the receiver, not RTS
- *
- * Revision 1.23 2000/01/24 17:46:19 johana
- * Wait for flush of DMA/FIFO when closing port.
- *
- * Revision 1.22 2000/01/20 18:10:23 johana
- * Added TIOCMGET ioctl to return modem status.
- * Implemented modem status/control that works with the extra signals
- * (DTR, DSR, RI,CD) as well.
- * 3 different modes supported:
- * ser0 on PB (Bundy), ser1 on PB (Lisa) and ser2 on PA (Bundy)
- * Fixed DEF_TX value that caused the serial transmitter pin (txd) to go to 0 when
- * closing the last filehandle, NASTY!.
- * Added break generation, not tested though!
- * Use IRQF_SHARED when request_irq() for ser2 and ser3 (shared with) par0 and par1.
- * You can't use them at the same time (yet..), but you can hopefully switch
- * between ser2/par0, ser3/par1 with the same kernel config.
- * Replaced some magic constants with defines
- *
- *
- */
```

[PATCH] CRISv10 serial driver rewrite

```
static char *serial_version = "$Revision: 1.25 $";
@@ -446,6 +30,7 @@ static char *serial_version = "$Revision: 1.25 $";

#include <asm/io.h>
#include <asm/irq.h>
+#include <asm/dma.h>
#include <asm/system.h>
#include <linux/delay.h>

@@ -454,8 +39,9 @@ static char *serial_version = "$Revision: 1.25 $";
/* non-arch dependent serial structures are in linux/serial.h */
#include <linux/serial.h>
/* while we keep our own stuff (struct e100_serial) in a local .h file */
-#include "serial.h"
+#include "crisv10.h"
#include <asm/fasttimer.h>
+#include <asm/arch/io_interface_mux.h>

#ifdef CONFIG_ETRAX_SERIAL_FAST_TIMER
#ifndef CONFIG_ETRAX_FAST_TIMER
@@ -504,18 +90,6 @@ struct tty_driver *serial_driver;
from eLinux */
#define SERIAL_HANDLE_EARLY_ERRORS

-/* Defined and used in n_tty.c, but we need it here as well */
-#define TTY_THRESHOLD_THROTTLE 128
-
-/* Due to buffersizes and threshold values, our SERIAL_DESCR_BUF_SIZE
- * must not be to high or flow control won't work if we leave it to the tty
- * layer so we have our own throttling in flush_to_flip
- * TTY_FLIPBUF_SIZE=512,
- * TTY_THRESHOLD_THROTTLE/UNTHROTTLE=128
- * BUF_SIZE can't be > 128
- */
-#define CRIS_BUF_SIZE 512
-
/* Currently 16 descriptors x 128 bytes = 2048 bytes */
#define SERIAL_DESCR_BUF_SIZE 256

@@ -588,13 +162,13 @@ unsigned long timer_data_to_ns(unsigned long timer_data);
static void change_speed(struct e100_serial *info);
static void rs_throttle(struct tty_struct * tty);
static void rs_wait_until_sent(struct tty_struct *tty, int timeout);
-static int rs_write(struct tty_struct * tty, int from_user,
- const unsigned char *buf, int count);
+static int rs_write(struct tty_struct *tty,
+ const unsigned char *buf, int count);
#ifdef CONFIG_ETRAX_RS485
-static int e100_write_rs485(struct tty_struct * tty, int from_user,
- const unsigned char *buf, int count);
```

[PATCH] CRISv10 serial driver rewrite

```
+static int e100_write_rs485(struct tty_struct *tty,
+ const unsigned char *buf, int count);
#endif
-static int get_lsr_info(struct e100_serial * info, unsigned int *value);
+static int get_lsr_info(struct e100_serial *info, unsigned int *value);

#define DEF_BAUD 115200 /* 115.2 kbit/s */
@@ -679,20 +253,39 @@ static struct e100_serial rs_table[] = {
.rx_ctrl = DEF_RX,
.tx_ctrl = DEF_TX,
.iseteop = 2,
+ .dma_owner = dma_ser0,
+ .io_if = if_serial_0,
#ifdef CONFIG_ETRAX_SERIAL_PORT0
.enabled = 1,
#ifdef CONFIG_ETRAX_SERIAL_PORT0_DMA6_OUT
.dma_out_enabled = 1,
+ .dma_out_nbr = SER0_TX_DMA_NBR,
+ .dma_out_irq_nbr = SER0_DMA_TX_IRQ_NBR,
+ .dma_out_irq_flags = IRQF_DISABLED,
+ .dma_out_irq_description = "serial 0 dma tr",
#else
.dma_out_enabled = 0,
+ .dma_out_nbr = UINT_MAX,
+ .dma_out_irq_nbr = 0,
+ .dma_out_irq_flags = 0,
+ .dma_out_irq_description = NULL,
#endif
#ifdef CONFIG_ETRAX_SERIAL_PORT0_DMA7_IN
.dma_in_enabled = 1,
+ .dma_in_nbr = SER0_RX_DMA_NBR,
+ .dma_in_irq_nbr = SER0_DMA_RX_IRQ_NBR,
+ .dma_in_irq_flags = IRQF_DISABLED,
+ .dma_in_irq_description = "serial 0 dma rec",
#else
- .dma_in_enabled = 0
+ .dma_in_enabled = 0,
+ .dma_in_nbr = UINT_MAX,
+ .dma_in_irq_nbr = 0,
+ .dma_in_irq_flags = 0,
+ .dma_in_irq_description = NULL,
#endif
#else
.enabled = 0,
+ .io_if_description = NULL,
.dma_out_enabled = 0,
.dma_in_enabled = 0
#endif
@@ -714,20 +307,42 @@ static struct e100_serial rs_table[] = {
.rx_ctrl = DEF_RX,
```

[PATCH] CRISv10 serial driver rewrite

```
.tx_ctrl = DEF_TX,
.iseteop = 3,
+ .dma_owner = dma_ser1,
+ .io_if = if_serial_1,
#ifdef CONFIG_ETRAX_SERIAL_PORT1
.enabled = 1,
+ .io_if_description = "ser1",
#ifdef CONFIG_ETRAX_SERIAL_PORT1_DMA8_OUT
.dma_out_enabled = 1,
+ .dma_out_nbr = SER1_TX_DMA_NBR,
+ .dma_out_irq_nbr = SER1_DMA_TX_IRQ_NBR,
+ .dma_out_irq_flags = IRQF_DISABLED,
+ .dma_out_irq_description = "serial 1 dma tr",
#else
.dma_out_enabled = 0,
+ .dma_out_nbr = UINT_MAX,
+ .dma_out_irq_nbr = 0,
+ .dma_out_irq_flags = 0,
+ .dma_out_irq_description = NULL,
#endif
#ifdef CONFIG_ETRAX_SERIAL_PORT1_DMA9_IN
.dma_in_enabled = 1,
+ .dma_in_nbr = SER1_RX_DMA_NBR,
+ .dma_in_irq_nbr = SER1_DMA_RX_IRQ_NBR,
+ .dma_in_irq_flags = IRQF_DISABLED,
+ .dma_in_irq_description = "serial 1 dma rec",
#else
- .dma_in_enabled = 0
+ .dma_in_enabled = 0,
+ .dma_in_enabled = 0,
+ .dma_in_nbr = UINT_MAX,
+ .dma_in_irq_nbr = 0,
+ .dma_in_irq_flags = 0,
+ .dma_in_irq_description = NULL,
#endif
#else
.enabled = 0,
+ .io_if_description = NULL,
+ .dma_in_irq_nbr = 0,
.dma_out_enabled = 0,
.dma_in_enabled = 0
#endif
@@ -748,20 +363,40 @@ static struct e100_serial rs_table[] = {
.rx_ctrl = DEF_RX,
.tx_ctrl = DEF_TX,
.iseteop = 0,
+ .dma_owner = dma_ser2,
+ .io_if = if_serial_2,
#ifdef CONFIG_ETRAX_SERIAL_PORT2
.enabled = 1,
+ .io_if_description = "ser2",
```

[PATCH] CRISv10 serial driver rewrite

```
#ifdef CONFIG_ETRAX_SERIAL_PORT2_DMA2_OUT
.dma_out_enabled = 1,
+ .dma_out_nbr = SER2_TX_DMA_NBR,
+ .dma_out_irq_nbr = SER2_DMA_TX_IRQ_NBR,
+ .dma_out_irq_flags = IRQF_DISABLED,
+ .dma_out_irq_description = "serial 2 dma tr",
#else
.dma_out_enabled = 0,
+ .dma_out_nbr = UINT_MAX,
+ .dma_out_irq_nbr = 0,
+ .dma_out_irq_flags = 0,
+ .dma_out_irq_description = NULL,
#endif
#ifdef CONFIG_ETRAX_SERIAL_PORT2_DMA3_IN
.dma_in_enabled = 1,
+ .dma_in_nbr = SER2_RX_DMA_NBR,
+ .dma_in_irq_nbr = SER2_DMA_RX_IRQ_NBR,
+ .dma_in_irq_flags = IRQF_DISABLED,
+ .dma_in_irq_description = "serial 2 dma rec",
#else
- .dma_in_enabled = 0
+ .dma_in_enabled = 0,
+ .dma_in_nbr = UINT_MAX,
+ .dma_in_irq_nbr = 0,
+ .dma_in_irq_flags = 0,
+ .dma_in_irq_description = NULL,
#endif
#else
.enabled = 0,
+ .io_if_description = NULL,
.dma_out_enabled = 0,
.dma_in_enabled = 0
#endif
@@ -782,20 +417,40 @@ static struct e100_serial rs_table[] = {
.rx_ctrl = DEF_RX,
.tx_ctrl = DEF_TX,
.iseteop = 1,
+ .dma_owner = dma_ser3,
+ .io_if = if_serial_3,
#ifdef CONFIG_ETRAX_SERIAL_PORT3
.enabled = 1,
+ .io_if_description = "ser3",
#ifdef CONFIG_ETRAX_SERIAL_PORT3_DMA4_OUT
.dma_out_enabled = 1,
+ .dma_out_nbr = SER3_TX_DMA_NBR,
+ .dma_out_irq_nbr = SER3_DMA_TX_IRQ_NBR,
+ .dma_out_irq_flags = IRQF_DISABLED,
+ .dma_out_irq_description = "serial 3 dma tr",
#else
.dma_out_enabled = 0,
+ .dma_out_nbr = UINT_MAX,
```

[PATCH] CRISv10 serial driver rewrite

```
+ .dma_out_irq_nbr = 0,
+ .dma_out_irq_flags = 0,
+ .dma_out_irq_description = NULL,
#endif
#ifdef CONFIG_ETRAX_SERIAL_PORT3_DMA5_IN
. dma_in_enabled = 1,
+ .dma_in_nbr = SER3_RX_DMA_NBR,
+ .dma_in_irq_nbr = SER3_DMA_RX_IRQ_NBR,
+ .dma_in_irq_flags = IRQF_DISABLED,
+ .dma_in_irq_description = "serial 3 dma rec",
#else
- .dma_in_enabled = 0
+ .dma_in_enabled = 0,
+ .dma_in_nbr = UINT_MAX,
+ .dma_in_irq_nbr = 0,
+ .dma_in_irq_flags = 0,
+ .dma_in_irq_description = NULL
#endif
#else
.enabled = 0,
+ .io_if_description = NULL,
. dma_out_enabled = 0,
. dma_in_enabled = 0
#endif
@@ -1416,12 +1071,11 @@ e100_dtr(struct e100_serial *info, int set)
{
unsigned long flags;

- save_flags(flags);
- cli();
+ local_irq_save(flags);
*e100_modem_pins[info->line].dtr_shadow &= ~mask;
*e100_modem_pins[info->line].dtr_shadow |= (set ? 0 : mask);
*e100_modem_pins[info->line].dtr_port = *e100_modem_pins[info->line].dtr_shadow;
- restore_flags(flags);
+ local_irq_restore(flags);
}

#ifdef SERIAL_DEBUG_IO
@@ -1440,12 +1094,11 @@ e100_rts(struct e100_serial *info, int set)
{
#ifdef CONFIG_SVINTO_SIM
unsigned long flags;
- save_flags(flags);
- cli();
+ local_irq_save(flags);
info->rx_ctrl &= ~E100_RTS_MASK;
info->rx_ctrl |= (set ? 0 : E100_RTS_MASK); /* RTS is active low */
info->port[REG_REC_CTRL] = info->rx_ctrl;
- restore_flags(flags);
+ local_irq_restore(flags);
```

[PATCH] CRISv10 serial driver rewrite

```
#ifdef SERIAL_DEBUG_IO
printk("ser%i rts %i\n", info->line, set);
#endif
@@ -1463,12 +1116,11 @@ e100_ri_out(struct e100_serial *info, int set)
unsigned char mask = e100_modem_pins[info->line].ri_mask;
unsigned long flags;

- save_flags(flags);
- cli();
+ local_irq_save(flags);
*e100_modem_pins[info->line].ri_shadow &= ~mask;
*e100_modem_pins[info->line].ri_shadow |= (set ? 0 : mask);
*e100_modem_pins[info->line].ri_port = *e100_modem_pins[info->line].ri_shadow;
- restore_flags(flags);
+ local_irq_restore(flags);
}
#endif
}
@@ -1481,12 +1133,11 @@ e100_cd_out(struct e100_serial *info, int set)
unsigned char mask = e100_modem_pins[info->line].cd_mask;
unsigned long flags;

- save_flags(flags);
- cli();
+ local_irq_save(flags);
*e100_modem_pins[info->line].cd_shadow &= ~mask;
*e100_modem_pins[info->line].cd_shadow |= (set ? 0 : mask);
*e100_modem_pins[info->line].cd_port = *e100_modem_pins[info->line].cd_shadow;
- restore_flags(flags);
+ local_irq_restore(flags);
}
#endif
}
@@ -1560,8 +1211,7 @@ static void e100_disable_txdma_channel(struct e100_serial *info)
/* Disable output DMA channel for the serial port in question
 * ( set to something other than serialX)
 */
- save_flags(flags);
- cli();
+ local_irq_save(flags);
DFLOW(DEBUG_LOG(info->line, "disable_txdma_channel %i\n", info->line));
if (info->line == 0) {
if ((genconfig_shadow & IO_MASK(R_GEN_CONFIG, dma6)) ==
@@ -1589,7 +1239,7 @@ static void e100_disable_txdma_channel(struct e100_serial *info)
}
}
*R_GEN_CONFIG = genconfig_shadow;
- restore_flags(flags);
+ local_irq_restore(flags);
}
```

[PATCH] CRISv10 serial driver rewrite

```
@@ -1597,8 +1247,7 @@ static void e100_enable_txdma_channel(struct e100_serial *info)
{
unsigned long flags;

- save_flags(flags);
- cli();
+ local_irq_save(flags);
DFLOW(DEBUG_LOG(info->line, "enable_txdma_channel %i\n", info->line));
/* Enable output DMA channel for the serial port in question */
if (info->line == 0) {
@@ -1615,7 +1264,7 @@ static void e100_enable_txdma_channel(struct e100_serial *info)
genconfig_shadow |= IO_STATE(R_GEN_CONFIG, dma4, serial3);
}
*R_GEN_CONFIG = genconfig_shadow;
- restore_flags(flags);
+ local_irq_restore(flags);
}

static void e100_disable_rxdma_channel(struct e100_serial *info)
@@ -1625,8 +1274,7 @@ static void e100_disable_rxdma_channel(struct e100_serial *info)
/* Disable input DMA channel for the serial port in question
* ( set to something other then serialX)
*/
- save_flags(flags);
- cli();
+ local_irq_save(flags);
if (info->line == 0) {
if ((genconfig_shadow & IO_MASK(R_GEN_CONFIG, dma7)) ==
IO_STATE(R_GEN_CONFIG, dma7, serial0)) {
@@ -1653,7 +1301,7 @@ static void e100_disable_rxdma_channel(struct e100_serial *info)
}
}
*R_GEN_CONFIG = genconfig_shadow;
- restore_flags(flags);
+ local_irq_restore(flags);
}

@@ -1661,8 +1309,7 @@ static void e100_enable_rxdma_channel(struct e100_serial *info)
{
unsigned long flags;

- save_flags(flags);
- cli();
+ local_irq_save(flags);
/* Enable input DMA channel for the serial port in question */
if (info->line == 0) {
genconfig_shadow &= ~IO_MASK(R_GEN_CONFIG, dma7);
@@ -1678,7 +1325,7 @@ static void e100_enable_rxdma_channel(struct e100_serial *info)
genconfig_shadow |= IO_STATE(R_GEN_CONFIG, dma5, serial3);
```

[PATCH] CRISv10 serial driver rewrite

```
}
*R_GEN_CONFIG = genconfig_shadow;
- restore_flags(flags);
+ local_irq_restore(flags);
}

#ifdef SERIAL_HANDLE_EARLY_ERRORS
@@ -1785,7 +1432,7 @@ e100_enable_rs485(struct tty_struct *tty, struct rs485_control *r)
}

static int
-e100_write_rs485(struct tty_struct *tty, int from_user,
+e100_write_rs485(struct tty_struct *tty,
const unsigned char *buf, int count)
{
struct e100_serial * info = (struct e100_serial *)tty->driver_data;
@@ -1798,7 +1445,7 @@ e100_write_rs485(struct tty_struct *tty, int from_user,
*/
info->rs485.enabled = 1;
/* rs_write now deals with RS485 if enabled */
- count = rs_write(tty, from_user, buf, count);
+ count = rs_write(tty, buf, count);
info->rs485.enabled = old_enabled;
return count;
}
@@ -1836,7 +1483,7 @@ rs_stop(struct tty_struct *tty)
unsigned long flags;
unsigned long xoff;

- save_flags(flags); cli();
+ local_irq_save(flags);
DFLOW(DEBUG_LOG(info->line, "XOFF rs_stop xmit %i\n",
CIRC_CNT(info->xmit.head,
info->xmit.tail, SERIAL_XMIT_SIZE)));
@@ -1848,7 +1495,7 @@ rs_stop(struct tty_struct *tty)
}

*((unsigned long *)&info->port[REG_XOFF]) = xoff;
- restore_flags(flags);
+ local_irq_restore(flags);
}
}

@@ -1860,7 +1507,7 @@ rs_start(struct tty_struct *tty)
unsigned long flags;
unsigned long xoff;

- save_flags(flags); cli();
+ local_irq_save(flags);
DFLOW(DEBUG_LOG(info->line, "XOFF rs_start xmit %i\n",
CIRC_CNT(info->xmit.head,
```

[PATCH] CRISv10 serial driver rewrite

```
info->xmit.tail, SERIAL_XMIT_SIZE));
@@ -1875,7 +1522,7 @@ rs_start(struct tty_struct *tty)
info->xmit.head != info->xmit.tail && info->xmit.buf)
e100_enable_serial_tx_ready_irq(info);

- restore_flags(flags);
+ local_irq_restore(flags);
}
}

@@ -2055,8 +1702,7 @@ static int serial_fast_timer_expired = 0;
static void flush_timeout_function(unsigned long data);
#define START_FLUSH_FAST_TIMER_TIME(info, string, usec) {\
unsigned long timer_flags; \
- save_flags(timer_flags); \
- cli(); \
+ local_irq_save(timer_flags); \
if (fast_timers[info->line].function == NULL) { \
serial_fast_timer_started++; \
TIMERD(DEBUG_LOG(info->line, "start_timer %i ", info->line)); \
@@ -2070,7 +1716,7 @@ static void flush_timeout_function(unsigned long data);
else { \
TIMERD(DEBUG_LOG(info->line, "timer %i already running\n", info->line)); \
} \
- restore_flags(timer_flags); \
+ local_irq_restore(timer_flags); \
}
#define START_FLUSH_FAST_TIMER(info, string) START_FLUSH_FAST_TIMER_TIME(info, string,
info->flush_time_usec)

@@ -2099,8 +1745,7 @@ append_rcv_buffer(struct e100_serial *info, struct etrax_rcv_buffer *buffer)
{
unsigned long flags;

- save_flags(flags);
- cli();
+ local_irq_save(flags);

if (!info->first_rcv_buffer)
info->first_rcv_buffer = buffer;
@@ -2113,7 +1758,7 @@ append_rcv_buffer(struct e100_serial *info, struct etrax_rcv_buffer *buffer)
if (info->rcv_cnt > info->max_rcv_cnt)
info->max_rcv_cnt = info->rcv_cnt;

- restore_flags(flags);
+ local_irq_restore(flags);
}

static int
@@ -2133,11 +1778,7 @@ add_char_and_flag(struct e100_serial *info, unsigned char data, unsigned char fl
info->icount.rx++;
```

[PATCH] CRISv10 serial driver rewrite

```
    } else {
struct tty_struct *tty = info->tty;
- *tty->flip.char_buf_ptr = data;
- *tty->flip.flag_buf_ptr = flag;
- tty->flip.flag_buf_ptr++;
- tty->flip.char_buf_ptr++;
- tty->flip.count++;
+ tty_insert_flip_char(tty, data, flag);
info->icount.rx++;
    }

@@ -2322,7 +1963,6 @@ start_receive(struct e100_serial *info)
*/
return;
#endif
- info->tty->flip.count = 0;
if (info->uses_dma_in) {
/* reset the input dma channel to be sure it works */

@@ -2484,32 +2124,20 @@ static void flush_to_flip_buffer(struct e100_serial *info)
{
struct tty_struct *tty;
struct etrax_recv_buffer *buffer;
- unsigned int length;
unsigned long flags;
- int max_flip_size;

- if (!info->first_recv_buffer)
- return;
-
- save_flags(flags);
- cli();
+ local_irq_save(flags);
+ tty = info->tty;

- if (!(tty = info->tty)) {
- restore_flags(flags);
+ if (!tty) {
+ local_irq_restore(flags);
return;
}

while ((buffer = info->first_recv_buffer) != NULL) {
unsigned int count = buffer->length;

- count = tty_buffer_request_room(tty, count);
- if (count == 0) /* Throttle ?? */
- break;
-
- if (count > 1)
- tty_insert_flip_strings(tty, buffer->buffer, count - 1);
```

[PATCH] CRISv10 serial driver rewrite

```
- tty_insert_flip_char(tty, buffer->buffer[count-1], buffer->error);
-
+ tty_insert_flip_string(tty, buffer->buffer, count);
info->recv_cnt -= count;

if (count == buffer->length) {
@@ -2525,18 +2153,9 @@ static void flush_to_flip_buffer(struct e100_serial *info)
if (!info->first_recv_buffer)
info->last_recv_buffer = NULL;

- restore_flags(flags);
-
- DFLIP(
- if (1) {
- DEBUG_LOG(info->line, "*** rxtot %i\n", info->icount.rx);
- DEBUG_LOG(info->line, "ldisc %lu\n", tty->ldisc.chars_in_buffer(tty));
- DEBUG_LOG(info->line, "room %lu\n", tty->ldisc.receive_room(tty));
- }
-
- );
+ local_irq_restore(flags);

- /* this includes a check for low-latency */
+ /* This includes a check for low-latency */
tty_flip_buffer_push(tty);
}

@@ -2679,21 +2298,7 @@ struct e100_serial * handle_ser_rx_interrupt_no_dma(struct e100_serial *info)
printk("!NO TTY!\n");
return info;
}
- if (tty->flip.count >= CRIS_BUF_SIZE - TTY_THRESHOLD_THROTTLE) {
- /* check TTY_THROTTLED first so it indicates our state */
- if (!test_and_set_bit(TTY_THROTTLED, &tty->flags)) {
- DFLOW(DEBUG_LOG(info->line, "rs_throttle flip.count: %i\n", tty->flip.count));
- rs_throttle(tty);
- }
- }
- if (tty->flip.count >= CRIS_BUF_SIZE) {
- DEBUG_LOG(info->line, "force FLIP! %i\n", tty->flip.count);
- tty->flip.work.func((void *) tty);
- if (tty->flip.count >= CRIS_BUF_SIZE) {
- DEBUG_LOG(info->line, "FLIP FULL! %i\n", tty->flip.count);
- return info; /* if TTY_DONT_FLIP is set */
- }
- }
+
+ /* Read data and status at the same time */
data_read = *((unsigned long *)&info->port[REG_DATA_STATUS32]);
more_data:
@@ -2746,27 +2351,26 @@ more_data:
```

[PATCH] CRISv10 serial driver rewrite

```
DEBUG_LOG(info->line, "EBRK %i\n", info->break_detected_cnt);
info->errorcode = ERRCODE_INSERT_BREAK;
} else {
+ unsigned char data = IO_EXTRACT(R_SERIAL0_READ,
+ data_in, data_read);
+ char flag = TTY_NORMAL;
if (info->errorcode == ERRCODE_INSERT_BREAK) {
- info->icount.brk++;
- *tty->flip.char_buf_ptr = 0;
- *tty->flip.flag_buf_ptr = TTY_BREAK;
- tty->flip.flag_buf_ptr++;
- tty->flip.char_buf_ptr++;
- tty->flip.count++;
+ struct tty_struct *tty = info->tty;
+ tty_insert_flip_char(tty, 0, flag);
info->icount.rx++;
}
- *tty->flip.char_buf_ptr = IO_EXTRACT(R_SERIAL0_READ, data_in, data_read);

if (data_read & IO_MASK(R_SERIAL0_READ, par_err)) {
info->icount.parity++;
- *tty->flip.flag_buf_ptr = TTY_PARITY;
+ flag = TTY_PARITY;
} else if (data_read & IO_MASK(R_SERIAL0_READ, overrun)) {
info->icount.overrun++;
- *tty->flip.flag_buf_ptr = TTY_OVERRUN;
+ flag = TTY_OVERRUN;
} else if (data_read & IO_MASK(R_SERIAL0_READ, framing_err)) {
info->icount.frame++;
- *tty->flip.flag_buf_ptr = TTY_FRAME;
+ flag = TTY_FRAME;
}
+ tty_insert_flip_char(tty, data, flag);
info->errorcode = 0;
}
info->break_detected_cnt = 0;
@@ -2782,16 +2386,14 @@ more_data:
log_int(rdpc(), 0, 0);
}
);
- *tty->flip.char_buf_ptr = IO_EXTRACT(R_SERIAL0_READ, data_in, data_read);
- *tty->flip.flag_buf_ptr = 0;
+ tty_insert_flip_char(tty,
+ IO_EXTRACT(R_SERIAL0_READ, data_in, data_read),
+ TTY_NORMAL);
} else {
DEBUG_LOG(info->line, "ser_rx int but no data_avail %08IX\n", data_read);
}

- tty->flip.flag_buf_ptr++;
```

[PATCH] CRISv10 serial driver rewrite

```
- tty->flip.char_buf_ptr++;
- tty->flip.count++;
info->icount.rx++;
data_read = *((unsigned long *)&info->port[REG_DATA_STATUS32]);
if (data_read & IO_MASK(R_SERIAL0_READ, data_avail)) {
@@ -2929,7 +2531,7 @@ static void handle_ser_tx_interrupt(struct e100_serial *info)
if (info->x_char) {
unsigned char rstat;
DFLOW(DEBUG_LOG(info->line, "tx_int: xchar 0x%02X\n", info->x_char));
- save_flags(flags); cli();
+ local_irq_save(flags);
rstat = info->port[REG_STATUS];
DFLOW(DEBUG_LOG(info->line, "stat %x\n", rstat));

@@ -2938,7 +2540,7 @@ static void handle_ser_tx_interrupt(struct e100_serial *info)
info->x_char = 0;
/* We must enable since it is disabled in ser_interrupt */
e100_enable_serial_tx_ready_irq(info);
- restore_flags(flags);
+ local_irq_restore(flags);
return;
}
if (info->uses_dma_out) {
@@ -2946,7 +2548,7 @@ static void handle_ser_tx_interrupt(struct e100_serial *info)
int i;
/* We only use normal tx interrupt when sending x_char */
DFLOW(DEBUG_LOG(info->line, "tx_int: xchar sent\n", 0));
- save_flags(flags); cli();
+ local_irq_save(flags);
rstat = info->port[REG_STATUS];
DFLOW(DEBUG_LOG(info->line, "stat %x\n", rstat));
e100_disable_serial_tx_ready_irq(info);
@@ -2959,7 +2561,7 @@ static void handle_ser_tx_interrupt(struct e100_serial *info)
nop();

*info->ocmdadr = IO_STATE(R_DMA_CH6_CMD, cmd, continue);
- restore_flags(flags);
+ local_irq_restore(flags);
return;
}
/* Normal char-by-char interrupt */
@@ -2973,7 +2575,7 @@ static void handle_ser_tx_interrupt(struct e100_serial *info)
}
DINTR2(DEBUG_LOG(info->line, "tx_int %c\n", info->xmit.buf[info->xmit.tail]));
/* Send a byte, rs485 timing is critical so turn of ints */
- save_flags(flags); cli();
+ local_irq_save(flags);
info->port[REG_TR_DATA] = info->xmit.buf[info->xmit.tail];
info->xmit.tail = (info->xmit.tail + 1) & (SERIAL_XMIT_SIZE-1);
info->icount.tx++;
@@ -2997,7 +2599,7 @@ static void handle_ser_tx_interrupt(struct e100_serial *info)
```

[PATCH] CRISv10 serial driver rewrite

```
/* We must enable since it is disabled in ser_interrupt */
e100_enable_serial_tx_ready_irq(info);
}
- restore_flags(flags);
+ local_irq_restore(flags);

if (CIRC_CNT(info->xmit.head,
info->xmit.tail,
@@ -3022,7 +2624,7 @@ ser_interrupt(int irq, void *dev_id)
int handled = 0;
static volatile unsigned long reentered_ready_mask = 0;

- save_flags(flags); cli();
+ local_irq_save(flags);
irq_mask1_rd = *R_IRQ_MASK1_RD;
/* First handle all rx interrupts with ints disabled */
info = rs_table;
@@ -3067,7 +2669,7 @@ ser_interrupt(int irq, void *dev_id)
/* Unblock the serial interrupt */
*R_VECT_MASK_SET = IO_STATE(R_VECT_MASK_SET, serial, set);

- sti();
+ local_irq_enable();
ready_mask = (1 << (8+1+2*0)); /* ser0 tr_ready */
info = rs_table;
for (i = 0; i < NR_PORTS; i++) {
@@ -3080,11 +2682,11 @@ ser_interrupt(int irq, void *dev_id)
ready_mask <<= 2;
}
/* handle_ser_tx_interrupt enables tr_ready interrupts */
- cli();
+ local_irq_disable();
/* Handle reentered TX interrupt */
irq_mask1_rd = reentered_ready_mask;
}
- cli();
+ local_irq_disable();
tx_started = 0;
} else {
unsigned long ready_mask;
@@ -3100,7 +2702,7 @@ ser_interrupt(int irq, void *dev_id)
}
}

- restore_flags(flags);
+ local_irq_restore(flags);
return IRQ_RETVAL(handled);
} /* ser_interrupt */
#endif
@@ -3121,11 +2723,13 @@ ser_interrupt(int irq, void *dev_id)
* them using rs_sched_event(), and they get done here.
```

[PATCH] CRISv10 serial driver rewrite

```
*/
static void
-do_softint(void *private_)
+do_softint(struct work_struct *work)
{
- struct e100_serial *info = (struct e100_serial *) private_;
+ struct e100_serial *info;
struct tty_struct *tty;

+ info = container_of(work, struct e100_serial, work);
+
tty = info->tty;
if (!tty)
return;
@@ -3145,13 +2749,12 @@ startup(struct e100_serial * info)
if (!xmit_page)
return -ENOMEM;

- save_flags(flags);
- cli();
+ local_irq_save(flags);

/* if it was already initialized, skip this */

if (info->flags & ASYNC_INITIALIZED) {
- restore_flags(flags);
+ local_irq_restore(flags);
free_page(xmit_page);
return 0;
}
@@ -3277,7 +2880,7 @@ startup(struct e100_serial * info)

info->flags |= ASYNC_INITIALIZED;

- restore_flags(flags);
+ local_irq_restore(flags);
return 0;
}

@@ -3328,8 +2931,7 @@ shutdown(struct e100_serial * info)
info->irq);
#endif

- save_flags(flags);
- cli(); /* Disable interrupts */
+ local_irq_save(flags);

if (info->xmit.buf) {
free_page((unsigned long)info->xmit.buf);
@@ -3353,7 +2955,7 @@ shutdown(struct e100_serial * info)
set_bit(TTY_IO_ERROR, &info->tty->flags);
```

[PATCH] CRISv10 serial driver rewrite

```
info->flags &= ~ASYNC_INITIALIZED;
- restore_flags(flags);
+ local_irq_restore(flags);
}

@@ -3411,7 +3013,6 @@ change_speed(struct e100_serial *info)
DBAUD(printk("using external baudrate: %lu\n", CONFIG_ETRAX_EXTERN_PB6CLK_FREQ/8));
info->baud = CONFIG_ETRAX_EXTERN_PB6CLK_FREQ/8;
}
- }
#endif
else
{
@@ -3445,8 +3046,7 @@ change_speed(struct e100_serial *info)

#ifdef CONFIG_SVINTO_SIM
/* start with default settings and then fill in changes */
- save_flags(flags);
- cli();
+ local_irq_save(flags);
/* 8 bit, no/even parity */
info->rx_ctrl &= ~(IO_MASK(R_SERIAL0_REC_CTRL, rec_bitnr) |
IO_MASK(R_SERIAL0_REC_CTRL, rec_par_en) |
@@ -3510,7 +3110,7 @@ change_speed(struct e100_serial *info)
}

*((unsigned long *)&info->port[REG_XOFF]) = xoff;
- restore_flags(flags);
+ local_irq_restore(flags);
#endif /* !CONFIG_SVINTO_SIM */

update_char_time(info);
@@ -3538,13 +3138,12 @@ rs_flush_chars(struct tty_struct *tty)

/* this protection might not exactly be necessary here */

- save_flags(flags);
- cli();
+ local_irq_save(flags);
start_transmit(info);
- restore_flags(flags);
+ local_irq_restore(flags);
}

-static int rs_raw_write(struct tty_struct * tty, int from_user,
+static int rs_raw_write(struct tty_struct *tty,
const unsigned char *buf, int count)
{
int c, ret = 0;
```

[PATCH] CRISv10 serial driver rewrite

```
@@ -3567,53 +3166,19 @@ static int rs_raw_write(struct tty_struct * tty, int from_user,
SIMCOUT(buf, count);
return count;
#endif
- save_flags(flags);
+ local_save_flags(flags);
DFLOW(DEBUG_LOG(info->line, "write count %i ", count));
DFLOW(DEBUG_LOG(info->line, "ldisc %i\n", tty->ldisc.chars_in_buffer(tty)));

- /* the cli/restore_flags pairs below are needed because the
- * DMA interrupt handler moves the info->xmit values. the memcpy
- * needs to be in the critical region unfortunately, because we
- * need to read xmit values, memcpy, write xmit values in one
- * atomic operation... this could perhaps be avoided by more clever
- * design.
+ /* The local_irq_disable/restore_flags pairs below are needed
+ * because the DMA interrupt handler moves the info->xmit values.
+ * the memcpy needs to be in the critical region unfortunately,
+ * because we need to read xmit values, memcpy, write xmit values
+ * in one atomic operation... this could perhaps be avoided by
+ * more clever design.
*/
- if (from_user) {
- mutex_lock(&tmp_buf_mutex);
- while (1) {
- int c1;
- c = CIRC_SPACE_TO_END(info->xmit.head,
- info->xmit.tail,
- SERIAL_XMIT_SIZE);
- if (count < c)
- c = count;
- if (c <= 0)
- break;
-
- c -= copy_from_user(tmp_buf, buf, c);
- if (!c) {
- if (!ret)
- ret = -EFAULT;
- break;
- }
- cli();
- c1 = CIRC_SPACE_TO_END(info->xmit.head,
- info->xmit.tail,
- SERIAL_XMIT_SIZE);
- if (c1 < c)
- c = c1;
- memcpy(info->xmit.buf + info->xmit.head, tmp_buf, c);
- info->xmit.head = ((info->xmit.head + c) &
- (SERIAL_XMIT_SIZE-1));
- restore_flags(flags);
```

[PATCH] CRISv10 serial driver rewrite

```
- buf += c;
- count -= c;
- ret += c;
- }
- mutex_unlock(&tmp_buf_mutex);
- } else {
- cli();
+ local_irq_disable();
while (count) {
c = CIRC_SPACE_TO_END(info->xmit.head,
info->xmit.tail,
@@ -3631,8 +3196,7 @@ static int rs_raw_write(struct tty_struct * tty, int from_user,
count -= c;
ret += c;
}
- restore_flags(flags);
- }
+ local_irq_restore(flags);

/* enable transmitter if not running, unless the tty is stopped
* this does not need IRQ protection since if tr_running == 0
@@ -3651,7 +3215,7 @@ static int rs_raw_write(struct tty_struct * tty, int from_user,
} /* raw_raw_write() */

static int
-rs_write(struct tty_struct * tty, int from_user,
+rs_write(struct tty_struct *tty,
const unsigned char *buf, int count)
{
#ifdef CONFIG_ETRAX_RS485
@@ -3678,7 +3242,7 @@ rs_write(struct tty_struct * tty, int from_user,
}
#endif /* CONFIG_ETRAX_RS485 */

- count = rs_raw_write(tty, from_user, buf, count);
+ count = rs_raw_write(tty, buf, count);

#ifdef CONFIG_ETRAX_RS485
if (info->rs485.enabled)
@@ -3746,10 +3310,9 @@ rs_flush_buffer(struct tty_struct *tty)
struct e100_serial *info = (struct e100_serial *)tty->driver_data;
unsigned long flags;

- save_flags(flags);
- cli();
+ local_irq_save(flags);
info->xmit.head = info->xmit.tail = 0;
- restore_flags(flags);
+ local_irq_restore(flags);

tty_wakeup(tty);
```

[PATCH] CRISv10 serial driver rewrite

```
}
@@ -3767,7 +3330,7 @@ static void rs_send_xchar(struct tty_struct *tty, char ch)
{
struct e100_serial *info = (struct e100_serial *)tty->driver_data;
unsigned long flags;
- save_flags(flags); cli();
+ local_irq_save(flags);
if (info->uses_dma_out) {
/* Put the DMA on hold and disable the channel */
*info->ocmdadr = IO_STATE(R_DMA_CH6_CMD, cmd, hold);
@@ -3784,7 +3347,7 @@ static void rs_send_xchar(struct tty_struct *tty, char ch)
DFLOW(DEBUG_LOG(info->line, "rs_send_xchar 0x%02X\n", ch));
info->x_char = ch;
e100_enable_serial_tx_ready_irq(info);
- restore_flags(flags);
+ local_irq_restore(flags);
}

/*
@@ -3996,21 +3559,61 @@ char *get_control_state_str(int MLines, char *s)
}
#endif

+static void
+rs_break(struct tty_struct *tty, int break_state)
+{
+ struct e100_serial *info = (struct e100_serial *)tty->driver_data;
+ unsigned long flags;
+
+ if (!info->port)
+ return;
+
+ local_irq_save(flags);
+ if (break_state == -1) {
+ /* Go to manual mode and set the txd pin to 0 */
+ /* Clear bit 7 (txd) and 6 (tr_enable) */
+ info->tx_ctrl &= 0x3F;
+ } else {
+ /* Set bit 7 (txd) and 6 (tr_enable) */
+ info->tx_ctrl |= (0x80 | 0x40);
+ }
+ info->port[REG_TR_CTRL] = info->tx_ctrl;
+ local_irq_restore(flags);
+}
+
static int
-get_modem_info(struct e100_serial * info, unsigned int *value)
+rs_tiocmset(struct tty_struct *tty, struct file *file,
+ unsigned int set, unsigned int clear)
{
- unsigned int result;
```

[PATCH] CRISv10 serial driver rewrite

```
- /* Polarity isn't verified */
-#if 0 /*def SERIAL_DEBUG_IO */
+ struct e100_serial *info = (struct e100_serial *)tty->driver_data;

- printk("get_modem_info: RTS: %i DTR: %i CD: %i RI: %i DSR: %i CTS: %i\n",
- E100_RTS_GET(info),
- E100_DTR_GET(info),
- E100_CD_GET(info),
- E100_RI_GET(info),
- E100_DSR_GET(info),
- E100_CTS_GET(info));
-#endif
+ if (clear & TIOCM_RTS)
+ e100_rts(info, 0);
+ if (clear & TIOCM_DTR)
+ e100_dtr(info, 0);
+ /* Handle FEMALE behaviour */
+ if (clear & TIOCM_RI)
+ e100_ri_out(info, 0);
+ if (clear & TIOCM_CD)
+ e100_cd_out(info, 0);
+
+ if (set & TIOCM_RTS)
+ e100_rts(info, 1);
+ if (set & TIOCM_DTR)
+ e100_dtr(info, 1);
+ /* Handle FEMALE behaviour */
+ if (set & TIOCM_RI)
+ e100_ri_out(info, 1);
+ if (set & TIOCM_CD)
+ e100_cd_out(info, 1);
+ return 0;
+ }
+
+static int
+rs_tiocmget(struct tty_struct *tty, struct file *file)
+{
+ struct e100_serial *info = (struct e100_serial *)tty->driver_data;
+ unsigned int result;

result =
(!E100_RTS_GET(info) ? TIOCM_RTS : 0)
@@ -4021,95 +3624,20 @@ get_modem_info(struct e100_serial * info, unsigned int *value)
| (!E100_CTS_GET(info) ? TIOCM_CTS : 0);

#ifdef SERIAL_DEBUG_IO
- printk("e100ser: modem state: %i 0x%08X\n", result, result);
+ printk(KERN_DEBUG "ser%i: modem state: %i 0x%08X\n",
+ info->line, result, result);
{
char s[100];
```

[PATCH] CRISv10 serial driver rewrite

```
get_control_state_str(result, s);
- printk("state: %s\n", s);
+ printk(KERN_DEBUG "state: %s\n", s);
}
#endif
- if (copy_to_user(value, &result, sizeof(int)))
- return -EFAULT;
- return 0;
-}
-
-
-static int
-set_modem_info(struct e100_serial * info, unsigned int cmd,
- unsigned int *value)
-{
- unsigned int arg;
+ return result;

- if (copy_from_user(&arg, value, sizeof(int)))
- return -EFAULT;
-
- switch (cmd) {
- case TIOCMCBIS:
- if (arg & TIOCM_RTS) {
- e100_rts(info, 1);
- }
- if (arg & TIOCM_DTR) {
- e100_dtr(info, 1);
- }
- /* Handle FEMALE behaviour */
- if (arg & TIOCM_RI) {
- e100_ri_out(info, 1);
- }
- if (arg & TIOCM_CD) {
- e100_cd_out(info, 1);
- }
- break;
- case TIOCMCBIC:
- if (arg & TIOCM_RTS) {
- e100_rts(info, 0);
- }
- if (arg & TIOCM_DTR) {
- e100_dtr(info, 0);
- }
- /* Handle FEMALE behaviour */
- if (arg & TIOCM_RI) {
- e100_ri_out(info, 0);
- }
- if (arg & TIOCM_CD) {
- e100_cd_out(info, 0);
```

[PATCH] CRISv10 serial driver rewrite

```
- }
- break;
- case TIOCMSET:
- e100_rts(info, arg & TIOCM_RTS);
- e100_dtr(info, arg & TIOCM_DTR);
- /* Handle FEMALE behaviour */
- e100_ri_out(info, arg & TIOCM_RI);
- e100_cd_out(info, arg & TIOCM_CD);
- break;
- default:
- return -EINVAL;
- }
- return 0;
}

-static void
-rs_break(struct tty_struct *tty, int break_state)
-{
- struct e100_serial * info = (struct e100_serial *)tty->driver_data;
- unsigned long flags;
-
- if (!info->port)
- return;
-
- save_flags(flags);
- cli();
- if (break_state == -1) {
- /* Go to manual mode and set the txd pin to 0 */
- info->tx_ctrl &= 0x3F; /* Clear bit 7 (txd) and 6 (tr_enable) */
- } else {
- info->tx_ctrl |= (0x80 | 0x40); /* Set bit 7 (txd) and 6 (tr_enable) */
- }
- info->port[REG_TR_CTRL] = info->tx_ctrl;
- restore_flags(flags);
-}

static int
rs_ioctl(struct tty_struct *tty, struct file * file,
unsigned int cmd, unsigned long arg)
@@ -4124,49 +3652,45 @@ rs_ioctl(struct tty_struct *tty, struct file * file,
}

switch (cmd) {
- case TIOCMGET:
- return get_modem_info(info, (unsigned int *) arg);
- case TIOCMIBIS:
- case TIOCMIBIC:
- case TIOCMSET:
- return set_modem_info(info, cmd, (unsigned int *) arg);
- case TIOCGSERIAL:
```

[PATCH] CRISv10 serial driver rewrite

```
- return get_serial_info(info,
- (struct serial_struct *) arg);
- case TIOCSSERIAL:
- return set_serial_info(info,
- (struct serial_struct *) arg);
- case TIOCSERGETLSR: /* Get line status register */
- return get_lsr_info(info, (unsigned int *) arg);
-
- case TIOCSERGSTRUCT:
- if (copy_to_user((struct e100_serial *) arg,
- info, sizeof(struct e100_serial)))
- return -EFAULT;
- return 0;
+ case TIOCGSERIAL:
+ return get_serial_info(info,
+ (struct serial_struct *) arg);
+ case TIOCSSERIAL:
+ return set_serial_info(info,
+ (struct serial_struct *) arg);
+ case TIOCSERGETLSR: /* Get line status register */
+ return get_lsr_info(info, (unsigned int *) arg);
+
+ case TIOCSERGSTRUCT:
+ if (copy_to_user((struct e100_serial *) arg,
+ info, sizeof(struct e100_serial)))
+ return -EFAULT;
+ return 0;

#if defined(CONFIG_ETRAX_RS485)
- case TIOCSERSETRS485:
- {
- struct rs485_control rs485ctrl;
- if (copy_from_user(&rs485ctrl, (struct rs485_control*)arg, sizeof(rs485ctrl)))
- return -EFAULT;
+ case TIOCSERSETRS485:
+ {
+ struct rs485_control rs485ctrl;
+ if (copy_from_user(&rs485ctrl, (struct rs485_control *)arg,
+ sizeof(rs485ctrl)))
+ return -EFAULT;

- return e100_enable_rs485(tty, &rs485ctrl);
- }
+ return e100_enable_rs485(tty, &rs485ctrl);
+ }

- case TIOCSERWRRS485:
- {
- struct rs485_write rs485wr;
- if (copy_from_user(&rs485wr, (struct rs485_write*)arg, sizeof(rs485wr)))
- return -EFAULT;
```

[PATCH] CRISv10 serial driver rewrite

```
+ case TIOCSERWRRS485:
+ {
+ struct rs485_write rs485wr;
+ if (copy_from_user(&rs485wr, (struct rs485_write *)arg,
+ sizeof(rs485wr)))
+ return -EFAULT;

- return e100_write_rs485(tty, 1, rs485wr.outc, rs485wr.outc_size);
- }
+ return e100_write_rs485(tty, rs485wr.outc, rs485wr.outc_size);
+ }
#endif

- default:
- return -ENOIOCTLCMD;
+ default:
+ return -ENOIOCTLCMD;
}
return 0;
}
@@ -4191,46 +3715,6 @@ rs_set_termios(struct tty_struct *tty, struct ktermios *old_termios)

}

-/* In debugport.c - register a console write function that uses the normal
- * serial driver
- */
-typedef int (*debugport_write_function)(int i, const char *buf, unsigned int len);
-
-extern debugport_write_function debug_write_function;
-
-static int rs_debug_write_function(int i, const char *buf, unsigned int len)
-{
- int cnt;
- int written = 0;
- struct tty_struct *tty;
- static int recurse_cnt = 0;
-
- tty = rs_table[i].tty;
- if (tty) {
- unsigned long flags;
- if (recurse_cnt > 5) /* We skip this debug output */
- return 1;
-
- local_irq_save(flags);
- recurse_cnt++;
- local_irq_restore(flags);
- do {
- cnt = rs_write(tty, 0, buf + written, len);
- if (cnt >= 0) {
- written += cnt;

```

[PATCH] CRISv10 serial driver rewrite

```
- buf += cnt;
- len -= cnt;
- } else
- len = cnt;
- } while(len > 0);
- local_irq_save(flags);
- recurse_cnt--;
- local_irq_restore(flags);
- return 1;
- }
- return 0;
-}
-
/*
* -----
* rs_close()
@@ -4252,11 +3736,10 @@ rs_close(struct tty_struct *tty, struct file * filp)

/* interrupts are disabled for this entire function */

- save_flags(flags);
- cli();
+ local_irq_save(flags);

if (tty_hung_up_p(filp)) {
- restore_flags(flags);
+ local_irq_restore(flags);
return;
}

@@ -4283,7 +3766,7 @@ rs_close(struct tty_struct *tty, struct file * filp)
info->count = 0;
}
if (info->count) {
- restore_flags(flags);
+ local_irq_restore(flags);
return;
}
info->flags |= ASYNC_CLOSING;
@@ -4337,7 +3820,7 @@ rs_close(struct tty_struct *tty, struct file * filp)
}
info->flags &= ~(ASYNC_NORMAL_ACTIVE|ASYNC_CLOSING);
wake_up_interruptible(&info->close_wait);
- restore_flags(flags);
+ local_irq_restore(flags);

/* port closed */

@@ -4359,6 +3842,30 @@ rs_close(struct tty_struct *tty, struct file * filp)
#endif
}
```

[PATCH] CRISv10 serial driver rewrite

```
#endif
+
+ /*
+ * Release any allocated DMA irq's.
+ */
+ if (info->dma_in_enabled) {
+   cris_free_dma(info->dma_in_nbr, info->dma_in_irq_description);
+   free_irq(info->dma_in_irq_nbr,
+   info);
+   info->uses_dma_in = 0;
+ #ifdef SERIAL_DEBUG_OPEN
+   printk(KERN_DEBUG "DMA irq '%s' freed\n",
+   info->dma_in_irq_description);
+ #endif
+ }
+ if (info->dma_out_enabled) {
+   free_irq(info->dma_out_irq_nbr,
+   info);
+   cris_free_dma(info->dma_out_nbr, info->dma_out_irq_description);
+   info->uses_dma_out = 0;
+ #ifdef SERIAL_DEBUG_OPEN
+   printk(KERN_DEBUG "DMA irq '%s' freed\n",
+   info->dma_out_irq_description);
+ #endif
+ }
+ }
+ }

/*
@@ -4434,7 +3941,7 @@ block_til_ready(struct tty_struct *tty, struct file * filp,
if (tty_hung_up_p(filp) ||
(info->flags & ASYNC_CLOSING)) {
if (info->flags & ASYNC_CLOSING)
- interruptible_sleep_on(&info->close_wait);
+ wait_event_interruptible(info->close_wait, 0);
#ifdef SERIAL_DO_RESTART
if (info->flags & ASYNC_HUP_NOTIFY)
return -EAGAIN;
@@ -4472,21 +3979,19 @@ block_til_ready(struct tty_struct *tty, struct file * filp,
printk("block_til_ready before block: ttyS%d, count = %d\n",
info->line, info->count);
#endif
- save_flags(flags);
- cli();
+ local_irq_save(flags);
if (!tty_hung_up_p(filp)) {
extra_count++;
info->count--;
}
- restore_flags(flags);
+ local_irq_restore(flags);
info->blocked_open++;
```

[PATCH] CRISv10 serial driver rewrite

```
while (1) {
- save_flags(flags);
- cli();
+ local_irq_save(flags);
/* assert RTS and DTR */
e100_rts(info, 1);
e100_dtr(info, 1);
- restore_flags(flags);
+ local_irq_restore(flags);
set_current_state(TASK_INTERRUPTIBLE);
if (tty_hung_up_p(filp) ||
!(info->flags & ASYNC_INITIALIZED)) {
@@ -4538,9 +4043,9 @@ rs_open(struct tty_struct *tty, struct file * filp)
struct e100_serial *info;
int retval, line;
unsigned long page;
+ int allocated_resources = 0;

/* find which port we want to open */
-
line = tty->index;

if (line < 0 || line >= NR_PORTS)
@@ -4581,7 +4086,7 @@ rs_open(struct tty_struct *tty, struct file * filp)
if (tty_hung_up_p(filp) ||
(info->flags & ASYNC_CLOSING)) {
if (info->flags & ASYNC_CLOSING)
- interruptible_sleep_on(&info->close_wait);
+ wait_event_interruptible(info->close_wait, 0);
#ifdef SERIAL_DO_RESTART
return ((info->flags & ASYNC_HUP_NOTIFY) ?
-EAGAIN : -ERESTARTSYS);
@@ -4591,19 +4096,118 @@ rs_open(struct tty_struct *tty, struct file * filp)
}

/*
+ * If DMA is enabled try to allocate the irq's.
+ */
+ if (info->count == 1) {
+ allocated_resources = 1;
+ if (info->dma_in_enabled) {
+ if (request_irq(info->dma_in_irq_nbr,
+ rec_interrupt,
+ info->dma_in_irq_flags,
+ info->dma_in_irq_description,
+ info)) {
+ printk(KERN_WARNING "DMA irq '%s' busy; "
+ "falling back to non-DMA mode\n",
+ info->dma_in_irq_description);
+ /* Make sure we never try to use DMA in */
+ /* for the port again. */
```

[PATCH] CRISv10 serial driver rewrite

```
+ info->dma_in_enabled = 0;
+ } else if (cris_request_dma(info->dma_in_nbr,
+ info->dma_in_irq_description,
+ DMA_VERBOSE_ON_ERROR,
+ info->dma_owner)) {
+ free_irq(info->dma_in_irq_nbr, info);
+ printk(KERN_WARNING "DMA '%s' busy; "
+ "falling back to non-DMA mode\n",
+ info->dma_in_irq_description);
+ /* Make sure we never try to use DMA in */
+ /* for the port again. */
+ info->dma_in_enabled = 0;
+ }
+#ifdef SERIAL_DEBUG_OPEN
+ else
+ printk(KERN_DEBUG "DMA irq '%s' allocated\n",
+ info->dma_in_irq_description);
+#endif
+ }
+ if (info->dma_out_enabled) {
+ if (request_irq(info->dma_out_irq_nbr,
+ tr_interrupt,
+ info->dma_out_irq_flags,
+ info->dma_out_irq_descri
```