

[PATCH 1/1] 8250_kgdb driver reworked

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2005-09/0310.html>

From: Tom Rini (trini_at_kernel.crashing.org)

Date: 09/01/05

Date: Thu, 1 Sep 2005 12:02:51 -0700

To: Kernel Mailing List <linux-kernel@vger.kernel.org>, Andrew Morton <akpm@osdl.org>

Andrew, I'll probably repost the whole series next week in hope you'll grab it for 2.6.13-mm2.

This is the I/O driver for any 8250-compatible UARTs. This also adds some small hooks into the normal serial core so that we can take away the uart and make sure only KGDB is trying to use it. We also make sure that if KGDB_8250 is enabled, SERIAL_8250_NR_UARTS is set to the default of 4. This is so that if we can always depend on that constant in the 8250 driver for giving our table a size. To make it easier (or in some cases, possible) to free up ports on a shared irq system, we add a line member to plat_serial8250_port.

Since the last time this was submitted, I've reworked the commandline param, kgdb8250=, to be of the form <io or mmio>,<address>,<baud rate>,<irq> for all platforms to always work as an override of the compiled in port. To compile in a port you either go by number and baud rate, or compile in the port information the same as you would pass it in on the command line. The config options are pared way down as well now (don't ask me why I forgot 'int' exists, I can't say...).

CC: Russell King <rmk@arm.linux.org.uk>, Bjorn Helgaas <bjorn.helgaas@hp.com>

This is the I/O driver for any 8250-compatible UARTs. This also adds some small hooks into the normal serial core so that we can take away the uart and make sure only KGDB is trying to use it. We also make sure that if KGDB_8250 is enabled, SERIAL_8250_NR_UARTS is set to the default of 4. This is so that if we can always depend on that constant in the 8250 driver for giving our table a size. To make it easier (or in some cases, possible) to free up ports on a shared irq system, we add a line member to plat_serial8250_port.

Since the last time this was submitted, I've reworked the commandline param, kgdb8250=, to be of the form <io or mmio>,<address>,<baud rate>,<irq> for all platforms to always work as an override of the compiled in port. To compile in a port you either go by number and baud rate, or compile in the port information the same as you would pass it in on the command line.

linux-2.6.13-trini/drivers/serial/8250.c		21 +
linux-2.6.13-trini/drivers/serial/8250.h		1

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```
linux-2.6.13-trini/drivers/serial/8250_kgdb.c | 538 ++++++
linux-2.6.13-trini/drivers/serial/Kconfig      | 2
linux-2.6.13-trini/drivers/serial/Makefile    | 1
linux-2.6.13-trini/lib/Kconfig.debug         | 60 ++
6 files changed, 621 insertions(+), 2 deletions(-)
diff -puN drivers/serial/8250.c~8250 drivers/serial/8250.c
--- linux-2.6.13/drivers/serial/8250.c~8250      2005-09-01 12:00:37.000000000 -0700
+++ linux-2.6.13-trini/drivers/serial/8250.c     2005-09-01 12:00:37.000000000 -0700
@@ -2516,6 +2516,27 @@ void serial8250_unregister_port(int line
 }
EXPORT_SYMBOL(serial8250_unregister_port);

+/**
+ * serial8250_unregister_by_port - remove a 16x50 serial port
+ * at runtime.
+ * @port: A &struct uart_port that describes the port to remove.
+ *
+ * Remove one serial port. This may not be called from interrupt
+ * context. We hand the port back to the our control.
+ */
+void serial8250_unregister_by_port(struct uart_port *port)
+{
+    struct uart_8250_port *uart;
+
+    down(&serial_sem);
+    uart = serial8250_find_match_or_unused(port);
+    up(&serial_sem);
+
+    if (uart)
+        serial8250_unregister_port(uart->port.line);
+}
+EXPORT_SYMBOL(serial8250_unregister_by_port);
+
+static int __init serial8250_init(void)
+{
+    int ret, i;
diff -puN drivers/serial/8250.h~8250 drivers/serial/8250.h
--- linux-2.6.13/drivers/serial/8250.h~8250      2005-09-01 12:00:37.000000000 -0700
+++ linux-2.6.13-trini/drivers/serial/8250.h     2005-09-01 12:00:37.000000000 -0700
@@ -19,6 +19,7 @@
 int serial8250_register_port(struct uart_port *);
 void serial8250_unregister_port(int line);
+void serial8250_unregister_by_port(struct uart_port *port);
 void serial8250_suspend_port(int line);
 void serial8250_resume_port(int line);

diff -puN drivers/serial/Kconfig~8250 drivers/serial/Kconfig
--- linux-2.6.13/drivers/serial/Kconfig~8250     2005-09-01 12:00:37.000000000 -0700
+++ linux-2.6.13-trini/drivers/serial/Kconfig     2005-09-01 12:00:37.000000000 -0700
@@ -87,7 +87,7 @@ config SERIAL_8250_ACPI
 config SERIAL_8250_NR_UARTS
     int "Maximum number of 8250/16550 serial ports"
     depends on SERIAL_8250
+    depends on SERIAL_8250 || KGDB_8250
     default "4"
     help
         Set this to the number of serial ports you want the driver
diff -L drivers/serial/kgdb_8250.c -puN /dev/null /dev/null
diff -puN drivers/serial/Makefile~8250 drivers/serial/Makefile
--- linux-2.6.13/drivers/serial/Makefile~8250    2005-09-01 12:00:37.000000000 -0700
```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```
+++ linux-2.6.13-trini/drivers/serial/Makefile 2005-09-01 12:00:37.000000000 -0700
@@ -57,3 +57,4 @@ obj-$(CONFIG_SERIAL_JSM) += jsm/
 obj-$(CONFIG_SERIAL_TXX9) += serial_txx9.o
 obj-$(CONFIG_SERIAL_VR41XX) += vr41xx_siu.o
 obj-$(CONFIG_SERIAL_SGI_IOC4) += ioc4_serial.o
+obj-$(CONFIG_KGDB_8250) += 8250_kgdb.o
diff -puN include/linux/kgdb.h~8250 include/linux/kgdb.h
diff -puN include/linux/serial_8250.h~8250 include/linux/serial_8250.h
diff -puN lib/Kconfig.debug~8250 lib/Kconfig.debug
--- linux-2.6.13/lib/Kconfig.debug~8250 2005-09-01 12:00:37.000000000 -0700
+++ linux-2.6.13-trini/lib/Kconfig.debug 2005-09-01 12:00:59.000000000 -0700
@@ -193,7 +193,7 @@ config KGDB_CONSOLE
 choice
     prompt "Method for KGDB communication"
     depends on KGDB
-     default KGDB_ONLY_MODULES
+     default KGDB_8250_NOMODULE
     default KGDB_MPSC if SERIAL_MPSC
     help
         There are a number of different ways in which you can communicate
@@ -212,6 +212,14 @@ config KGDB_ONLY_MODULES
     Use only kernel modules to configure KGDB I/O after the
     kernel is booted.

+config KGDB_8250_NOMODULE
+ bool "KGDB: On generic serial port (8250)"
+ select KGDB_8250
+ help
+     Uses generic serial port (8250) to communicate with the host
+     GDB. This is independent of the normal (SERIAL_8250) driver
+     for this chipset.
+
 config KGDB_MPSC
     bool "KGDB on MV64x60 MPSC"
     depends on SERIAL_MPSC
@@ -219,4 +227,54 @@ config KGDB_MPSC
     Uses a Marvell GT64260B or MV64x60 Multi-Purpose Serial
     Controller (MPSC) channel. Note that the GT64260A is not
     supported.
+
+ endchoice
+
+config KGDB_8250
+ tristate "KGDB: On generic serial port (8250)" if !KGDB_8250_NOMODULE
+ depends on m && KGDB_ONLY_MODULES
+ help
+     Uses generic serial port (8250) to communicate with the host
+     GDB. This is independent of the normal (SERIAL_8250) driver
+     for this chipset.
+
+config KGDB_SIMPLE_SERIAL
+ bool "Simple selection of KGDB serial port"
+ depends on KGDB_8250
+ default y
+ help
+     If you say Y here, you will only have to pick the baud rate
+     and port number that you wish to use for KGDB. Note that this
+     only works on architectures that register known serial ports
+     early on. If you say N, you will have to provide, either here
+     or on the command line, the type (I/O or MMIO), IRQ and
+     address to use. If in doubt, say Y.
+
```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```
+config KGDB_BAUDRATE
+   int "Debug serial port baud rate"
+   depends on (KGDB_8250 && KGDB_SIMPLE_SERIAL)
+   default "115200"
+   help
+       gdb and the kernel stub need to agree on the baud rate to be
+       used.  Standard rates from 9600 to 115200 are allowed, and this
+       may be overridden via the commandline.
+
+config KGDB_PORT_NUM
+   int "Serial port number for KGDB"
+   range 0 1 if KGDB_MPSC
+   range 0 3
+   depends on (KGDB_8250 && KGDB_SIMPLE_SERIAL) || KGDB_MPSC
+   default "1"
+   help
+       Pick the port number (0 based) for KGDB to use.
+
+config KGDB_8250_CONF_STRING
+   string "Configuration string for KGDB"
+   depends on KGDB_8250 && !KGDB_SIMPLE_SERIAL
+   default "io,2f8,115200,3" if X86
+   help
+       The format of this string should be <io or
+       mem>,<address>,<baud rate>,<irq>.  For example, to use the
+       serial port on an i386 box located at 0x2f8 and 115200 baud
+       on IRQ 3 at use:
+       io,2f8,115200,3
diff -puN /dev/null drivers/serial/8250_kgdb.c
--- /dev/null      2005-08-29 18:56:12.476393250 -0700
+++ linux-2.6.13-trini/drivers/serial/8250_kgdb.c      2005-09-01 12:00:37.000000000 -0700
@@ -0,0 +1,538 @@
+/*
+ * 8250 interface for kgdb.
+ *
+ * This is a merging of many different drivers, and all of the people have
+ * had an impact in some form or another:
+ *
+ * 2004-2005 (c) MontaVista Software, Inc.
+ * 2005 (c) Wind River Systems, Inc.
+ *
+ * Amit Kale <amitkale@emsyssoft.com>, David Grothe <dave@gcom.com>,
+ * Scott Foehner <sfoehner@engr.sgi.com>, George Anzinger <george@mvista.com>,
+ * Robert Walsh <rjwalsh@durables.org>, wangdi <>wangdi@clusterfs.com>,
+ * San Mehat, Tom Rini <trini@mvista.com>,
+ * Jason Wessel <jason.wessel@windriver.com>
+ */
+
+#include <linux/config.h>
+#include <linux/kernel.h>
+#include <linux/init.h>
+#include <linux/kgdb.h>
+#include <linux/interrupt.h>
+#include <linux/tty.h>
+#include <linux/serial.h>
+#include <linux/serial_reg.h>
+#include <linux/serialP.h>
+#include <linux/ioport.h>
+
+#include <asm/io.h>
+#include <asm/serial.h>          /* For BASE_BAUD and SERIAL_PORT_DFNS */
+
```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```
+#include "8250.h"
+
+#define GDB_BUF_SIZE    512    /* power of 2, please */
+
+MODULE_DESCRIPTION("KGDB driver for the 8250");
+MODULE_LICENSE("GPL");
+/* These will conflict with early_param otherwise. */
+#ifdef CONFIG_KGDB_8250_MODULE
+static char config[256];
+module_param_string(kgdb8250, config, 256, 0);
+MODULE_PARM_DESC(kgdb8250,
+                 " kgdb8250=<io or mem>,<address>,<baud rate>,<irq>\n");
+static struct kgdb_io local_kgdb_io_ops;
+#endif
+                               /* CONFIG_KGDB_8250_MODULE */
+
+/* Speed of the UART. */
+static int kgdb8250_baud;
+/* Index of the UART, matches ttySX naming typically. */
+static int kgdb8250_ttyS;
+
+/* Flag for if we need to call request_mem_region */
+static int kgdb8250_needs_request_mem_region;
+
+static char kgdb8250_buf[GDB_BUF_SIZE];
+static atomic_t kgdb8250_buf_in_cnt;
+static int kgdb8250_buf_out_inx;
+
+/* Old-style serial definitions, if existant, and a counter. */
+#ifdef CONFIG_KGDB_SIMPLE_SERIAL
+static int __initdata should_copy_rs_table = 1;
+static struct serial_state old_rs_table[] __initdata = {
+#ifdef SERIAL_PORT_DFNS
+    SERIAL_PORT_DFNS
+#endif
+};
+#endif
+
+/* Our internal table of UARTS. */
+#define UART_NR          CONFIG_SERIAL_8250_NR_UARTS
+static struct uart_port kgdb8250_ports[UART_NR];
+
+/* Macros to easily get what we want from kgdb8250_ports[kgdb8250_ttyS] */
+#define CURRENTPORT      kgdb8250_ports[kgdb8250_ttyS]
+#define KGDB8250_IRQ     CURRENTPORT.irq
+#define KGDB8250_REG_SHIFT    CURRENTPORT.regshift
+
+/* Base of the UART. */
+static void *kgdb8250_addr;
+
+/* Forward declarations. */
+static int kgdb8250_init(void);
+static int __init kgdb_init_io(void);
+static int __init kgdb8250_opt(char *str);
+
+/*
+ * Wait until the interface can accept a char, then write it.
+ */
+static void kgdb_put_debug_char(u8 chr)
+{
+    while (!(ioread8(kgdb8250_addr + (UART_LSR << KGDB8250_REG_SHIFT)) &
+           UART_LSR_THRE)) ;
+}
```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```
+     iowrite8(chr, kgdb8250_addr + (UART_TX << KGDB8250_REG_SHIFT));
+}
+
+/*
+ * Get a byte from the hardware data buffer and return it
+ */
+static int read_data_bfr(void)
+{
+     char it = ioread8(kgdb8250_addr + (UART_LSR << KGDB8250_REG_SHIFT));
+
+     if (it & UART_LSR_DR)
+         return ioread8(kgdb8250_addr + (UART_RX << KGDB8250_REG_SHIFT));
+
+     /*
+      * If we have a framing error assume somebody messed with
+      * our uart. Reprogram it and send '-' both ways...
+      */
+     if (it & 0xc) {
+         kgdb8250_init();
+         kgdb_put_debug_char('-');
+         return '-';
+     }
+
+     return -1;
+}
+
+/*
+ * Get a char if available, return -1 if nothing available.
+ * Empty the receive buffer first, then look at the interface hardware.
+ */
+static int kgdb_get_debug_char(void)
+{
+     int retchr;
+
+     /* intr routine has q'd chars */
+     if (atomic_read(&kgdb8250_buf_in_cnt) != 0) {
+         retchr = kgdb8250_buf[kgdb8250_buf_out_inx++];
+         kgdb8250_buf_out_inx &= (GDB_BUF_SIZE - 1);
+         atomic_dec(&kgdb8250_buf_in_cnt);
+         return retchr;
+     }
+
+     do {
+         retchr = read_data_bfr();
+     } while (retchr < 0);
+
+     return retchr;
+}
+
+/*
+ * This is the receiver interrupt routine for the GDB stub.
+ * All that we need to do is verify that the interrupt happened on the
+ * line we're in charge of.  If this is true, schedule a breakpoint and
+ * return.
+ */
+static irqreturn_t
+kgdb8250_interrupt(int irq, void *dev_id, struct pt_regs *regs)
+{
+     char iir;
+
+     if (irq != KGDB8250_IRQ)
+         return IRQ_NONE;
```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```
+ /*
+  * If there is some other CPU in KGDB then this is a
+  * spurious interrupt. so return without even checking a byte
+  */
+ if (atomic_read(&debugger_active))
+     return IRQ_NONE;
+
+ iir = ioread8(kgdb8250_addr + (UART_IIR << KGDB8250_REG_SHIFT));
+ if (iir & UART_IIR_RDI) {
+     if (kgdb_io_ops.read_char != kgdb_get_debug_char) {
+         /* Throw away the data if another I/O routine
+          * is active.
+          */
+         char it = ioread8(kgdb8250_addr +
+                           (UART_LSR << KGDB8250_REG_SHIFT));
+         if (it & UART_LSR_DR)
+             ioread8(kgdb8250_addr +
+                     (UART_RX << KGDB8250_REG_SHIFT));
+     } else
+         breakpoint();
+ }
+
+ return IRQ_HANDLED;
+}
+
+/*
+ * Returns:
+ * 0 on success, 1 on failure.
+ */
+static int kgdb8250_init(void)
+{
+    unsigned cval;
+    int cflag = CREAD | HUPCL | CLOCAL | CS8;
+    char ier = UART_IER_RDI;
+    unsigned int base_baud;
+
+    base_baud = CURRENTPORT.uartclk ? CURRENTPORT.uartclk / 16 : BASE_BAUD;
+
+    /*
+     * Now construct a cflag setting.
+     */
+    switch (kgdb8250_baud) {
+    case 1200:
+        cflag |= B1200;
+        break;
+    case 2400:
+        cflag |= B2400;
+        break;
+    case 4800:
+        cflag |= B4800;
+        break;
+    case 19200:
+        cflag |= B19200;
+        break;
+    case 38400:
+        cflag |= B38400;
+        break;
+    case 57600:
+        cflag |= B57600;
+        break;
+    case 115200:
+        cflag |= B115200;
```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```

+         break;
+     default:
+         kgdb8250_baud = 9600;
+         /* Fall through */
+     case 9600:
+         cflag |= B9600;
+         break;
+     }
+
+     /*
+      *     Divisor, bytesize and parity
+      *
+      */
+
+     cval = cflag & (CSIZE | CSTOPB);
+     cval >= 4;
+
+     /* Disable UART interrupts, set DTR and RTS high and set speed. */
+#if defined(CONFIG_ARCH_OMAP1510)
+     /* Workaround to enable 115200 baud on OMAP1510 internal ports */
+     if (cpu_is_omap1510() && is_omap_port((void *)kgdb8250_addr)) {
+         if (kgdb8250_baud == 115200) {
+             base_baud = 1;
+             kgdb8250_baud = 1;
+             iowrite8(1, kgdb8250_addr +
+                 (UART_OMAP_OSC_12M_SEL << KGDB8250_REG_SHIFT));
+         } else {
+             iowrite8(0, kgdb8250_addr +
+                 (UART_OMAP_OSC_12M_SEL << KGDB8250_REG_SHIFT));
+         }
+     }
+#endif
+     /* set DLAB */
+     iowrite8(cval | UART_LCR_DLAB, kgdb8250_addr +
+         (UART_LCR << KGDB8250_REG_SHIFT));
+
+     /* LS */
+     iowrite8(base_baud / kgdb8250_baud & 0xff, kgdb8250_addr +
+         (UART_DLL << KGDB8250_REG_SHIFT));
+
+     /* MS */
+     iowrite8(base_baud / kgdb8250_baud >> 8, kgdb8250_addr +
+         (UART_DLM << KGDB8250_REG_SHIFT));
+
+     /* reset DLAB */
+     iowrite8(cval, kgdb8250_addr + (UART_LCR << KGDB8250_REG_SHIFT));
+
+     /*
+      * XScale-specific bits that need to be set
+      */
+     if (CURRENTPORT.type == PORT_XSCALE)
+         ier |= UART_IER_UUE | UART_IER_RTOIE;
+
+     /* turn on interrupts */
+     iowrite8(ier, kgdb8250_addr + (UART_IER << KGDB8250_REG_SHIFT));
+     iowrite8(UART_MCR_OUT2 | UART_MCR_DTR | UART_MCR_RTS,
+         kgdb8250_addr + (UART_MCR << KGDB8250_REG_SHIFT));
+
+     /*
+      *     If we read 0xff from the LSR, there is no UART here.
+      */
+     if (ioread8(kgdb8250_addr + (UART_LSR << KGDB8250_REG_SHIFT)) == 0xff)
+         return -1;
+     return 0;
+ }

```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```

+
+/*
+ * Copy the old serial_state table to our uart_port table if we haven't
+ * had values specifically configured in. We need to make sure this only
+ * happens once.
+ */
+static void __init kgdb8250_copy_rs_table(void)
+{
+#ifdef CONFIG_KGDB_SIMPLE_SERIAL
+    int i;
+
+    if (!should_copy_rs_table)
+        return;
+
+    for (i = 0; i < ARRAY_SIZE(old_rs_table); i++) {
+        kgdb8250_ports[i].iobase = old_rs_table[i].port;
+        kgdb8250_ports[i].irq = irq_canonicalize(old_rs_table[i].irq);
+        kgdb8250_ports[i].uartclk = old_rs_table[i].baud_base * 16;
+        kgdb8250_ports[i].membase = old_rs_table[i].iomem_base;
+        kgdb8250_ports[i].iotype = old_rs_table[i].io_type;
+        kgdb8250_ports[i].regshift = old_rs_table[i].iomem_reg_shift;
+        kgdb8250_ports[i].line = i;
+    }
+
+    should_copy_rs_table = 0;
+#endif
+}
+
+/*
+ * Hookup our IRQ line now that it is safe to do so, after we grab any
+ * memory regions we might need to. If we haven't been initialized yet,
+ * go ahead and copy the old_rs_table in.
+ */
+static void __init kgdb8250_late_init(void)
+{
+    /* Try and copy the old_rs_table. */
+    kgdb8250_copy_rs_table();
+
+#if defined(CONFIG_SERIAL_8250) || defined(CONFIG_SERIAL_8250_MODULE)
+    /* Take the port away from the main driver. */
+    serial8250_unregister_by_port(&CURRENTPORT);
+
+    /* Now reinit the port as the above has disabled things. */
+    kgdb8250_init();
+#endif
+
+    /* We may need to call request_mem_region() first. */
+    if (kgdb8250_needs_request_mem_region)
+        request_mem_region(CURRENTPORT.mapbase,
+                            8 << KGDB8250_REG_SHIFT, "kgdb");
+    if (request_irq(KGDB8250_IRQ, kgdb8250_interrupt, SA_SHIRQ,
+                   "GDB-stub", &CURRENTPORT) < 0)
+        printk(KERN_ERR "KGDB failed to request the serial IRQ (%d)\n",
+               KGDB8250_IRQ);
+}
+
+static __init int kgdb_init_io(void)
+{
+    /* Give us the basic table of uarts. */
+    kgdb8250_copy_rs_table();
+
+#ifdef CONFIG_KGDB_8250_MODULE
+    if (strlen(config)) {

```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```

+         if (kgdb8250_opt(config))
+             return -EINVAL;
+     } else {
+         printk(KERN_ERR "kgdb8250: argument error, usage: "
+             "kgdb8250=<io or mem>,<address>,<baud rate>" ",<irq>\n");
+         return -EINVAL;
+     }
+ #endif
+         /* CONFIG_KGDB_8250_MODULE */
+ #ifdef CONFIG_KGDB_SIMPLE_SERIAL
+     kgdb8250_baud = CONFIG_KGDB_BAUDRATE;
+     kgdb8250_ttyS = CONFIG_KGDB_PORT_NUM;
+ #else
+     if (kgdb8250_opt(CONFIG_KGDB_8250_CONF_STRING))
+         return -EINVAL;
+ #endif
+
+     /* Internal driver setup. */
+     switch (CURRENTPORT.iotype) {
+     case UPIO_MEM:
+         if (CURRENTPORT.mapbase)
+             kgdb8250_needs_request_mem_region = 1;
+         if (CURRENTPORT.flags & UPF_IOREMAP) {
+             CURRENTPORT.membase = ioremap(CURRENTPORT.mapbase,
+                 8 << KGDB8250_REG_SHIFT);
+             if (!CURRENTPORT.membase)
+                 return -EIO; /* Failed. */
+         }
+         kgdb8250_addr = CURRENTPORT.membase;
+         break;
+     case UPIO_PORT:
+     default:
+         kgdb8250_addr = ioport_map(CURRENTPORT.iobase,
+             8 << KGDB8250_REG_SHIFT);
+         if (!kgdb8250_addr)
+             return -EIO; /* Failed. */
+     }
+
+     if (kgdb8250_init() == -1) {
+         printk(KERN_ERR "kgdb8250: init failed\n");
+         return -EIO;
+     }
+ #ifdef CONFIG_KGDB_8250_MODULE
+     /* Attach the kgdb irq. When this is built into the kernel, it
+     * is called as a part of late_init sequence.
+     */
+     kgdb8250_late_init();
+     if (kgdb_register_io_module(&local_kgdb_io_ops))
+         return -EINVAL;
+
+     printk(KERN_INFO "kgdb8250: debugging enabled\n");
+ #endif
+     /* CONFIG_KGD_8250_MODULE */
+
+     return 0;
+ }
+
+ #ifdef CONFIG_KGDB_8250_MODULE
+ /* If it is a module the kgdb_io_ops should be a static which
+ * is passed to the KGDB I/O initialization
+ */
+ static struct kgdb_io local_kgdb_io_ops = {
+ #else
+     /* ! CONFIG_KGDB_8250_MODULE */
+ struct kgdb_io kgdb_io_ops = {

```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```
+#endif                                /* ! CONFIG_KGD_8250_MODULE */
+   .read_char = kgdb_get_debug_char,
+   .write_char = kgdb_put_debug_char,
+   .init = kgdb_init_io,
+   .late_init = kgdb8250_late_init,
+};
+
+/**
+ *   kgdb8250_get_ttyS - Return the index of the UART used by kgdb,
+ *   matches ttySX naming.
+ */
+int kgdb8250_get_ttyS(void)
+{
+   return kgdb8250_ttyS;
+}
+
+/**
+ *   kgdb8250_add_port - Define a serial port for use with KGDB
+ *   @i: The index of the port being added
+ *   @serial_req: The &struct uart_port describing the port
+ *
+ *   On platforms where we must register the serial device
+ *   dynamically, this is the best option if a platform also normally
+ *   calls early_serial_setup().
+ */
+void __init kgdb8250_add_port(int i, struct uart_port *serial_req)
+{
+   /* Make sure we've got the built-in data before we override. */
+   kgdb8250_copy_rs_table();
+
+   /* Copy the whole thing over. */
+   memcpy(&kgdb8250_ports[i], serial_req, sizeof(struct uart_port));
+}
+
+/**
+ *   kgdb8250_add_platform_port - Define a serial port for use with KGDB
+ *   @i: The index of the port being added
+ *   @p: The &struct plat_serial8250_port describing the port
+ *
+ *   On platforms where we must register the serial device
+ *   dynamically, this is the best option if a platform normally
+ *   handles uart setup with an array of &struct plat_serial8250_port.
+ */
+void __init kgdb8250_add_platform_port(int i, struct plat_serial8250_port *p)
+{
+   /* Make sure we've got the built-in data before we override. */
+   kgdb8250_copy_rs_table();
+
+   kgdb8250_ports[i].iobase = p->iobase;
+   kgdb8250_ports[i].membase = p->membase;
+   kgdb8250_ports[i].irq = p->irq;
+   kgdb8250_ports[i].uartclk = p->uartclk;
+   kgdb8250_ports[i].regshift = p->regshift;
+   kgdb8250_ports[i].iotype = p->iotype;
+   kgdb8250_ports[i].flags = p->flags;
+   kgdb8250_ports[i].mapbase = p->mapbase;
+}
+
+/**
+ * Syntax for this cmdline option is:
+ * kgdb8250=<io or mem>,<address>,<baud rate>,<irq>"
+ */
```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```

+static int __init kgdb8250_opt(char *str)
+{
+    /* Fill out or overwrite and use the first entry. */
+    kgdb8250_ttyS = 0;
+
+    if (!strncmp(str, "io", 2)) {
+        CURRENTPORT.iotype = UPIO_PORT;
+        str += 2;
+    } else if (!strncmp(str, "mmio", 4)) {
+        CURRENTPORT.iotype = UPIO_MEM;
+        str += 4;
+    } else
+        goto errout;
+
+    if (*str != ',')
+        goto errout;
+    str++;
+
+    if (CURRENTPORT.iotype == UPIO_PORT)
+        CURRENTPORT.iobase = simple_strtoul(str, &str, 16);
+    else
+        CURRENTPORT.membase =
+            (unsigned char *)simple_strtoul(str, &str, 16);
+
+    if (*str != ',')
+        goto errout;
+    str++;
+
+    kgdb8250_baud = simple_strtoul(str, &str, 10);
+    if (!kgdb8250_baud)
+        goto errout;
+
+    if (*str != ',')
+        goto errout;
+    str++;
+
+    CURRENTPORT.irq = simple_strtoul(str, &str, 10);
+
+#ifdef CONFIG_KGDB_SIMPLE_SERIAL
+    should_copy_rs_table = 0;
+#endif
+
+    return 0;
+
+errout:
+    printk(KERN_ERR "Invalid syntax for option kgdb8250=\n");
+    return 1;
+}
+
+#ifdef CONFIG_KGDB_8250_MODULE
+static void cleanup_kgdb8250(void)
+{
+    kgdb_unregister_io_module(&local_kgdb_io_ops);
+
+    /* Clean up the irq and memory */
+    free_irq(KGDB8250_IRQ, &CURRENTPORT);
+
+    if (kgdb8250_needs_request_mem_region)
+        release_mem_region(CURRENTPORT.mapbase,
+                            8 << KGDB8250_REG_SHIFT);
+
+    /* Hook up the serial port back to what it was previously
+     * hooked up to.

```

Linux-Kernel: [PATCH 1/1] 8250_kgdb driver reworked

```
+      */
+#if defined(CONFIG_SERIAL_8250) || defined(CONFIG_SERIAL_8250_MODULE)
+      /* Give the port back to the 8250 driver. */
+      serial8250_register_port(&CURRENTPORT);
+#endif
+}
+
+module_init(kgdb_init_io);
+module_exit(cleanup_kgdb8250);
+#else      /* ! CONFIG_KGDB_8250_MODULE */
+early_param("kgdb8250", kgdb8250_opt);
+#endif      /* ! CONFIG_KGDB_8250_MODULE */
```

--

Tom Rini

<http://gate.crashing.org/~trini/>

-

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