

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-10/msg05267.html>

- *From:* Vitaliy Ivanov <vitalivanov@xxxxxxxxxx>
 - *Date:* Tue, 16 Oct 2007 16:54:49 +0300
-

Willy,

On Mon, 2007-10-15 at 01:39, Willy Tarreau wrote:

That's what I've seen. I can propose you something (unless someone else raises his hand saying "no") : you update your patch with a short description of what the hardware module is supposed to be used for, and you accept to step up as the maintainer for this backport, which will imply that you put your name and mail in the MAINTAINERS file. That way, if you're the only user, nobody will be annoyed, and if there are other users and some of them have problems, I don't waste my time on something I don't know at all. If you agree with this deal (which I think is fair), then I'm willing to merge your patch into 2.4.36-pre.

It's completely fair. I spent some time on lkml and I liked it. I've a device and I can check/correct any issue we'll have with it(hope there won't be any:)). So it's OK for me to be a maintainer for this driver.

Here is final patch with all issues corrected.

Again, comments are welcomed.

Vitaliy

adutux is a simple Linux device driver for ADU boards from Ontrak Control Systems. The adutux driver exposes standard open/close/read/write API's to the user application.

Signed-off-by: Vitaliy Ivanov <vitalivanov@xxxxxxxxxx>

Tested-by: Vitaliy Ivanov <vitalivanov@xxxxxxxxxx>

diff -uprN -X dontdiff KERNELS/linux-2.4.35.3/Documentation/Configure.help

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

linux-2.4.35.3.build/Documentation/Configure.help

--- KERNELS/linux-2.4.35.3/Documentation/Configure.help 2007-09-24 01:02:58.000000000 +0300

+++ linux-2.4.35.3.build/Documentation/Configure.help 2007-10-14 19:19:06.000000000 +0300

@@ -16123,6 +16123,17 @@ CONFIG_USB_RIO500

The module will be called rio500.o. If you want to compile it as a module, say M here and read <file:Documenatation/modules.txt>.

+Ontrak Control Systems ADU devices support

+CONFIG_USB_ADUTUX

+ Say Y here if you want to connect ADU 100/200 series devices to your computer's USB port.

+ Details at: <<http://www.ontrak.net/products.htm#Table%205>>

+

+ This code is also available as a module (= code which can be inserted in and removed from the running kernel whenever you want).

+ The module will be called adutux.o. If you want to compile it as a module, say M here and read <file:Documenatation/modules.txt>.

+

Auerswald device support

CONFIG_USB_AUERSWALD

Say Y here if you want to connect an Auerswald USB ISDN Device

diff -uprN -X dontdiff KERNELS/linux-2.4.35.3/drivers/usb/adutux.c

linux-2.4.35.3.build/drivers/usb/adutux.c

--- KERNELS/linux-2.4.35.3/drivers/usb/adutux.c 1970-01-01 03:00:00.000000000 +0300

+++ linux-2.4.35.3.build/drivers/usb/adutux.c 2007-10-16 16:27:41.000000000 +0300

@@ -0,0 +1,1043 @@

+/*

+ * This is a 2.4.* kernel version of adutux driver that was originally created by John Homppi for 2.6.* kernels.

+ *

+ * Copyright (c) 2007 Vitaliy Ivanov (vitalivanov@xxxxxxxxxx)

+ *

+ * Original note:

+ *

+ * adutux - driver for ADU devices from Ontrak Control Systems

+ * This is an experimental driver. Use at your own risk.

+ * This driver is not supported by Ontrak Control Systems.

+ *

+ * Copyright (c) 2003 John Homppi (SCO, leave this notice here)

+ *

+ * This program is free software; you can redistribute it and/or

+ * modify it under the terms of the GNU General Public License as

+ * published by the Free Software Foundation; either version 2 of

+ * the License, or (at your option) any later version.

+ *

+ * derived from the Lego USB Tower driver 0.56:

+ * Copyright (c) 2003 David Glance <davidgsf@xxxxxxxxxxxxxxxx>

+ * 2001 Juergen Stuber <stuber@xxxxxxxx>

+ * that was derived from USB Skeleton driver - 0.5

+ * Copyright (c) 2001 Greg Kroah-Hartman (greg@xxxxxxxx)

+ *

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ */
+
+#include <linux/kernel.h>
+#include <linux/errno.h>
+#include <linux/init.h>
+#include <linux/slab.h>
+#include <linux/module.h>
+#include <linux/moduleparam.h>
+#include <linux/devfs_fs_kernel.h>
+#include <linux/usb.h>
+#include <asm/uaccess.h>
+
+#ifdef CONFIG_USB_DEBUG
+static int debug = 5;
+#else
+static int debug = 1;
+#endif
+
+/* Use our own dbg macro */
+#undef dbg
+#define dbg(lvl, format, arg...) \
+do { \
+ if (debug >= lvl) \
+ printk(KERN_DEBUG __FILE__ " : " format "\n", ## arg); \
+} while (0)
+
+
+/* Version Information */
+#define DRIVER_VERSION "v0.0.13"
+#define DRIVER_AUTHOR "John Homppi, Vitaliy Ivanov"
+#define DRIVER_DESC "adutux (see www.ontrak.net)"
+
+/* Module parameters */
+module_param(debug, int, S_IRUGO | S_IWUSR);
+MODULE_PARM_DESC(debug, "Debug enabled or not");
+
+/* Define these values to match your device */
+#define ADU_VENDOR_ID 0x0a07
+#define ADU_PRODUCT_ID 0x0064
+
+/* table of devices that work with this driver */
+static struct usb_device_id device_table [] = {
+ { USB_DEVICE(ADU_VENDOR_ID, ADU_PRODUCT_ID) }, /* ADU100 */
+ { USB_DEVICE(ADU_VENDOR_ID, ADU_PRODUCT_ID+20) }, /* ADU120 */
+ { USB_DEVICE(ADU_VENDOR_ID, ADU_PRODUCT_ID+30) }, /* ADU130 */
+ { USB_DEVICE(ADU_VENDOR_ID, ADU_PRODUCT_ID+100) }, /* ADU200 */
+ { USB_DEVICE(ADU_VENDOR_ID, ADU_PRODUCT_ID+108) }, /* ADU208 */
+ { USB_DEVICE(ADU_VENDOR_ID, ADU_PRODUCT_ID+118) }, /* ADU218 */
+ { } /* Terminating entry */
+};
+
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+MODULE_DEVICE_TABLE(usb, device_table);
+
+#define ADU_MINOR_BASE 67
+
+/* we can have up to this number of device plugged in at once */
+#define MAX_DEVICES 16
+
+#define COMMAND_TIMEOUT (2*HZ) /* 60 second timeout for a command */
+
+/* Structure to hold all of our device specific stuff */
+struct adu_device {
+ struct semaphore sem; /* locks this structure */
+ struct usb_device* udev; /* save off the usb device pointer */
+ //struct usb_interface* interface;
+ devfs_handle_t devfs; /* devfs device node */
+ unsigned char minor; /* the starting minor number for this device */
+
+ int open_count; /* number of times this port has been opened */
+
+ char* read_buffer_primary;
+ int read_buffer_length;
+ char* read_buffer_secondary;
+ int secondary_head;
+ int secondary_tail;
+ spinlock_t buflock;
+
+ wait_queue_head_t read_wait;
+ wait_queue_head_t write_wait;
+
+ char* interrupt_in_buffer;
+ struct usb_endpoint_descriptor* interrupt_in_endpoint;
+ struct urb* interrupt_in_urb;
+ int read_urb_finished;
+
+ char* interrupt_out_buffer;
+ struct usb_endpoint_descriptor* interrupt_out_endpoint;
+ struct urb* interrupt_out_urb;
+};
+
+/* the global usb devfs handle */
+extern devfs_handle_t usb_devfs_handle;
+
+/* local function prototypes */
+static ssize_t adu_read(struct file *file, char *buffer, size_t count, loff_t *ppos);
+static ssize_t adu_write(struct file *file, const char *buffer, size_t count, loff_t *ppos);
+static void adu_delete(struct adu_device *dev);
+static int adu_open(struct inode *inode, struct file *file);
+static int adu_release(struct inode *inode, struct file *file);
+static int adu_release_internal(struct adu_device *dev);
+static void adu_interrupt_in_callback(struct urb *urb);
+static void adu_interrupt_out_callback(struct urb *urb);
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+static void *adu_probe(struct usb_device *dev, unsigned int ifnum, const struct usb_device_id *id);
+static void adu_disconnect(struct usb_device *dev, void *ptr);
+
+/* array of pointers to our devices that are currently connected */
+static struct adu_device *minor_table[MAX_DEVICES];
+
+/* lock to protect the minor_table structure */
+static DECLARE_MUTEX(minor_table_mutex);
+
+/* file operations needed when we register this driver */
+static struct file_operations adu_fops = {
+ owner: THIS_MODULE,
+ read: adu_read,
+ write: adu_write,
+ open: adu_open,
+ release: adu_release,
+};
+
+/* usb specific object needed to register this driver with the usb subsystem */
+static struct usb_driver adu_driver = {
+ name: "adutux",
+ probe: adu_probe,
+ disconnect: adu_disconnect,
+ fops: &adu_fops,
+ minor: ADU_MINOR_BASE,
+ id_table: device_table,
+};
+
+/**
+ * adu_debug_data
+ */
+static void adu_debug_data(int level, const char *function, int size,
+ const unsigned char *data)
+{
+ int i;
+
+ if (debug < level)
+ return;
+
+ printk(KERN_DEBUG __FILE__: %s - length = %d, data = ",
+ function, size);
+ for (i = 0; i < size; ++i)
+ printk("%.2x ", data[i]);
+ printk("\n");
+}
+
+/**
+ * adu_abort_transfers
+ * aborts transfers and frees associated data structures
+ */
+static void adu_abort_transfers(struct adu_device *dev)
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+{
+ dbg(2, " %s : enter", __FUNCTION__);
+
+ if (dev == NULL) {
+ dbg(1, " %s : dev is null", __FUNCTION__);
+ goto exit;
+ }
+
+ if (dev->udev == NULL) {
+ dbg(1, " %s : udev is null", __FUNCTION__);
+ goto exit;
+ }
+
+ /* shutdown transfer */
+ usb_unlink_urb(dev->interrupt_in_urb);
+ usb_unlink_urb(dev->interrupt_out_urb);
+
+exit:
+ dbg(2, " %s : leave", __FUNCTION__);
+ }
+
+/**
+ * adu_delete
+ */
+static void adu_delete(struct adu_device *dev)
+{
+ dbg(2, "%s enter", __FUNCTION__);
+
+
+ minor_table[dev->minor] = NULL;
+ adu_abort_transfers(dev);
+
+
+ /* free data structures */
+ if (dev->interrupt_in_urb != NULL) {
+ usb_free_urb(dev->interrupt_in_urb);
+ }
+ if (dev->interrupt_out_urb != NULL) {
+ usb_free_urb(dev->interrupt_out_urb);
+ }
+ if (dev->read_buffer_primary != NULL) {
+ kfree(dev->read_buffer_primary);
+ }
+ if (dev->read_buffer_secondary != NULL) {
+ kfree(dev->read_buffer_secondary);
+ }
+ if (dev->interrupt_in_buffer != NULL) {
+ kfree(dev->interrupt_in_buffer);
+ }
+ if (dev->interrupt_out_buffer != NULL) {
+ kfree(dev->interrupt_out_buffer);
+ }
+ kfree(dev);
+}
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+
+ dbg(2, "%s : leave", __FUNCTION__);
+}
+
+/**
+ * adu_interrupt_in_callback
+ */
+static void adu_interrupt_in_callback(struct urb *urb)
+{
+ struct adu_device *dev = urb->context;
+
+ dbg(4, " %s : enter, status %d", __FUNCTION__, urb->status);
+ adu_debug_data(5, __FUNCTION__, urb->actual_length,
+ urb->transfer_buffer);
+
+ spin_lock(&dev->buflock);
+
+ if (urb->status != 0) {
+ if ((urb->status != -ENOENT) && (urb->status != -ECONNRESET) &&
+ (urb->status != -ESHUTDOWN)) {
+ dbg(1, " %s : nonzero status received: %d",
+ __FUNCTION__, urb->status);
+ }
+ goto exit;
+ }
+
+ if (urb->actual_length > 0 && dev->interrupt_in_buffer[0] != 0x00) {
+ if (dev->read_buffer_length <
+ (4 * le16_to_cpu(dev->interrupt_in_endpoint->wMaxPacketSize)) -
+ (urb->actual_length)) {
+ memcpy (dev->read_buffer_primary +
+ dev->read_buffer_length,
+ dev->interrupt_in_buffer, urb->actual_length);
+
+ dev->read_buffer_length += urb->actual_length;
+ dbg(2, " %s reading %d ", __FUNCTION__,
+ urb->actual_length);
+ } else {
+ dbg(1, " %s : read_buffer overflow", __FUNCTION__);
+ }
+ }
+
+exit:
+ dev->read_urb_finished = 1;
+ spin_unlock(&dev->buflock);
+ /* always wake up so we recover from errors */
+ wake_up_interruptible(&dev->read_wait);
+ adu_debug_data(5, __FUNCTION__, urb->actual_length,
+ urb->transfer_buffer);
+ dbg(4, " %s : leave, status %d", __FUNCTION__, urb->status);
+}
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+
+/**
+ * adu_interrupt_out_callback
+ */
+static void adu_interrupt_out_callback(struct urb *urb)
+{
+ struct adu_device *dev = urb->context;
+
+ dbg(4, " %s : enter, status %d", __FUNCTION__, urb->status);
+ adu_debug_data(5, __FUNCTION__, urb->actual_length, urb->transfer_buffer);
+
+ if (urb->status != 0) {
+ if ((urb->status != -ENOENT) &&
+ (urb->status != -ECONNRESET)) {
+ dbg(1, " %s : nonzero status received: %d",
+ __FUNCTION__, urb->status);
+ }
+ goto exit;
+ }
+
+ wake_up_interruptible(&dev->write_wait);
+exit:
+
+ adu_debug_data(5, __FUNCTION__, urb->actual_length,
+ urb->transfer_buffer);
+ dbg(4, " %s : leave, status %d", __FUNCTION__, urb->status);
+}
+
+/**
+ * adu_open
+ */
+static int adu_open(struct inode *inode, struct file *file)
+{
+ struct adu_device *dev = NULL;
+ int subminor;
+ int retval = 0;
+
+ dbg(2, "%s : enter", __FUNCTION__);
+
+ subminor = MINOR(inode->i_rdev) - ADU_MINOR_BASE;
+ if ((subminor < 0) ||
+ (subminor >= MAX_DEVICES)) {
+ retval = -ENODEV;
+ goto exit;
+ }
+
+ /* lock our minor table and get our local data for this minor */
+ down(&minor_table_mutex);
+ dev = minor_table[subminor];
+ if (dev == NULL) {
+ retval = -ENODEV;

```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ goto unlock_exit;
+ }
+
+ /* lock this device */
+ down(&dev->sem);
+
+ /* increment our usage count for the device */
+ ++dev->open_count;
+ dbg(2, "%s : open count %d", __FUNCTION__, dev->open_count);
+
+ /* check that nobody else is using the device */
+ /* NOTE: the cleanup code will decrement the count to 1 */
+ if (dev->open_count > 1) {
+   retval = -EBUSY;
+   /* unlock this device */
+   up(&dev->sem);
+   goto error;
+ }
+
+ /* save device in the file's private structure */
+ file->private_data = dev;
+
+ /* initialize in direction */
+ dev->read_buffer_length = 0;
+
+ /* fixup first read by having urb waiting for it */
+ FILL_INT_URB(dev->interrupt_in_urb,
+ dev->udev,
+ usb_rcvintpipe(dev->udev, dev->interrupt_in_endpoint->bEndpointAddress),
+ dev->interrupt_in_buffer,
+ le16_to_cpu(dev->interrupt_in_endpoint->wMaxPacketSize),
+ adu_interrupt_in_callback,
+ dev,
+ 0);
+
+ /* dev->interrupt_in_urb->transfer_flags |= URB_ASYNC_UNLINK; */
+ dev->read_urb_finished = 0;
+ retval = usb_submit_urb(dev->interrupt_in_urb);
+ if (retval != 0) {
+   err("Couldn't submit interrupt_in_urb");
+   /* we ignore failure, just notify about it */
+   /*goto error;*/
+ }
+ /* end of fixup for first read */
+
+ /* unlock this device */
+ up(&dev->sem);
+ goto unlock_exit;
+
+error:
+ adu_release_internal(dev);
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+
+unlock_exit:
+ /* unlock the minor table */
+ up(&minor_table_mutex);
+
+exit:
+ dbg(2, " %s : leave, return value %d", __FUNCTION__, retval);
+
+ return retval;
+}
+
+/**
+ * adu_release_internal
+ */
+static int adu_release_internal(struct adu_device *dev)
+{
+ int retval = 0;
+
+ dbg(2, " %s : enter", __FUNCTION__);
+
+ if (dev->udev == NULL) {
+ /* the device was unplugged before the file was released */
+ adu_delete(dev);
+ goto exit;
+ }
+
+ /* lock this device */
+ down(&dev->sem);
+
+ /* decrement our usage count for the device */
+ --dev->open_count;
+ dbg(2, " %s : open count %d", __FUNCTION__, dev->open_count);
+ if (dev->open_count <= 0) {
+ adu_abort_transfers(dev);
+ dev->open_count = 0;
+ }
+
+ /* unlock this device */
+ up(&dev->sem);
+
+exit:
+ dbg(2, " %s : leave", __FUNCTION__);
+ return retval;
+}
+
+/**
+ * adu_release
+ */
+static int adu_release(struct inode *inode, struct file *file)
+{
+ struct adu_device *dev = NULL;
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ int retval = 0;
+
+ dbg(2, " %s : enter", __FUNCTION__);
+
+ if (file == NULL) {
+ dbg(1, " %s : file is NULL", __FUNCTION__);
+ retval = -ENODEV;
+ goto exit;
+ }
+
+ dev = file->private_data;
+
+ if (dev == NULL) {
+ dbg(1, " %s : object is NULL", __FUNCTION__);
+ retval = -ENODEV;
+ goto exit;
+ }
+
+ /* lock our minor table */
+ down(&minor_table_mutex);
+
+ /* lock our device */
+ down(&dev->sem);
+
+ if (dev->open_count <= 0) {
+ dbg(1, " %s : device not opened", __FUNCTION__);
+ up(&dev->sem);
+ up(&minor_table_mutex);
+ retval = -ENODEV;
+ goto exit;
+ }
+
+ /* unlocking device */
+ up(&dev->sem);
+
+ /* do the work */
+ retval = adu_release_internal(dev);
+
+ up(&minor_table_mutex);
+
+exit:
+ dbg(2, " %s : leave, return value %d", __FUNCTION__, retval);
+ return retval;
+}
+
+/**
+ * adu_read
+ */
+static ssize_t adu_read(struct file *file, __user char *buffer, size_t count,
+ loff_t *ppos)
+{
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ struct adu_device *dev;
+ size_t bytes_read = 0;
+ size_t bytes_to_read = count;
+ int i;
+ int retval = 0;
+ int timeout = 0;
+ int should_submit = 0;
+ unsigned long flags;
+ DECLARE_WAITQUEUE(wait, current);
+
+ dbg(2, " %s : enter, count = %Zd, file=%p", __FUNCTION__, count, file);
+
+ dev = file->private_data;
+ dbg(2, " %s : dev=%p", __FUNCTION__, dev);
+ /* lock this object */
+ if (down_interruptible(&dev->sem))
+ return -ERESTARTSYS;
+
+ /* verify that the device wasn't unplugged */
+ if (dev->udev == NULL) {
+ retval = -ENODEV;
+ err("No device or device unplugged %d", retval);
+ goto exit;
+ }
+
+ /* verify that some data was requested */
+ if (count == 0) {
+ dbg(1, " %s : read request of 0 bytes", __FUNCTION__);
+ goto exit;
+ }
+
+ timeout = COMMAND_TIMEOUT;
+ dbg(2, " %s : about to start looping", __FUNCTION__);
+ while (bytes_to_read) {
+ int data_in_secondary = dev->secondary_tail - dev->secondary_head;
+ dbg(2, " %s : while, data_in_secondary=%d, status=%d",
+ __FUNCTION__, data_in_secondary,
+ dev->interrupt_in_urb->status);
+
+ if (data_in_secondary) {
+ /* drain secondary buffer */
+ int amount = bytes_to_read < data_in_secondary ? bytes_to_read : data_in_secondary;
+ i = copy_to_user(buffer, dev->read_buffer_secondary+dev->secondary_head, amount);
+ if (i < 0) {
+ retval = -EFAULT;
+ goto exit;
+ }
+ dev->secondary_head += (amount - i);
+ bytes_read += (amount - i);
+ bytes_to_read -= (amount - i);
+ if (i) {
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ retval = bytes_read ? bytes_read : -EFAULT;
+ goto exit;
+ }
+ } else {
+ /* we check the primary buffer */
+ spin_lock_irqsave (&dev->buflock, flags);
+ if (dev->read_buffer_length) {
+ /* we secure access to the primary */
+ char *tmp;
+ dbg(2, " %s : swap, read_buffer_length = %d",
+ __FUNCTION__, dev->read_buffer_length);
+ tmp = dev->read_buffer_secondary;
+ dev->read_buffer_secondary = dev->read_buffer_primary;
+ dev->read_buffer_primary = tmp;
+ dev->secondary_head = 0;
+ dev->secondary_tail = dev->read_buffer_length;
+ dev->read_buffer_length = 0;
+ spin_unlock_irqrestore(&dev->buflock, flags);
+ /* we have a free buffer so use it */
+ should_submit = 1;
+ } else {
+ /* even the primary was empty - we may need to do IO */
+ if (dev->interrupt_in_urb->status == -EINPROGRESS) {
+ /* somebody is doing IO */
+ spin_unlock_irqrestore(&dev->buflock, flags);
+ dbg(2, " %s : submitted already", __FUNCTION__);
+ } else {
+ /* we must initiate input */
+ dbg(2, " %s : initiate input", __FUNCTION__);
+ dev->read_urb_finished = 0;
+
+ +
+ + FILL_INT_URB(dev->interrupt_in_urb,
+ + dev->udev,
+ + usb_rcvintpipe(dev->udev, dev->interrupt_in_endpoint->bEndpointAddress),
+ + dev->interrupt_in_buffer,
+ + le16_to_cpu(dev->interrupt_in_endpoint->wMaxPacketSize),
+ + adu_interrupt_in_callback,
+ + dev,
+ + 0);
+
+ +
+ + retval = usb_submit_urb(dev->interrupt_in_urb);
+ + if (!retval) {
+ + spin_unlock_irqrestore(&dev->buflock, flags);
+ + dbg(2, " %s : submitted OK", __FUNCTION__);
+ + } else {
+ + if (retval == -ENOMEM) {
+ + retval = bytes_read ? bytes_read : -ENOMEM;
+ + }
+ + spin_unlock_irqrestore(&dev->buflock, flags);
+ + dbg(2, " %s : submit failed", __FUNCTION__);
+ + goto exit;

```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ }
+ }
+
+ /* we wait for I/O to complete */
+ set_current_state(TASK_INTERRUPTIBLE);
+ add_wait_queue(&dev->read_wait, &wait);
+ if (!dev->read_urb_finished)
+ timeout = schedule_timeout(COMMAND_TIMEOUT);
+ else
+ set_current_state(TASK_RUNNING);
+ remove_wait_queue(&dev->read_wait, &wait);
+
+ if (timeout <= 0) {
+ dbg(2, " %s : timeout", __FUNCTION__);
+ retval = bytes_read ? bytes_read : -ETIMEDOUT;
+ goto exit;
+ }
+
+ if (signal_pending(current)) {
+ dbg(2, " %s : signal pending", __FUNCTION__);
+ retval = bytes_read ? bytes_read : -EINTR;
+ goto exit;
+ }
+ }
+ }
+ }
+
+ retval = bytes_read;
+ /* if the primary buffer is empty then use it */
+ if (should_submit && !dev->interrupt_in_urb->status==EINPROGRESS) {
+ FILL_INT_URB(dev->interrupt_in_urb,
+ dev->udev,
+ usb_rcvintpipe(dev->udev, dev->interrupt_in_endpoint->bEndpointAddress),
+ dev->interrupt_in_buffer,
+ le16_to_cpu(dev->interrupt_in_endpoint->wMaxPacketSize),
+ adu_interrupt_in_callback,
+ dev,
+ 0);
+
+ /* dev->interrupt_in_urb->transfer_flags |= URB_ASYNC_UNLINK; */
+ dev->read_urb_finished = 0;
+ usb_submit_urb(dev->interrupt_in_urb);
+ /* we ignore failure */
+ }
+
+exit:
+ /* unlock the device */
+ up(&dev->sem);
+
+ dbg(2, " %s : leave, return value %d", __FUNCTION__, retval);
+ return retval;
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+}
+
+/**
+ * adu_write
+ */
+static ssize_t adu_write(struct file *file, const __user char *buffer,
+ size_t count, loff_t *ppos)
+{
+ struct adu_device *dev;
+ size_t bytes_written = 0;
+ size_t bytes_to_write;
+ size_t buffer_size;
+ int retval = 0;
+ int timeout = 0;
+
+ dbg(2, " %s : enter, count = %Zd", __FUNCTION__, count);
+
+ dev = file->private_data;
+
+ /* lock this object */
+ down_interruptible(&dev->sem);
+
+ /* verify that the device wasn't unplugged */
+ if (dev->udev == NULL) {
+ retval = -ENODEV;
+ err("No device or device unplugged %d", retval);
+ goto exit;
+ }
+
+ /* verify that we actually have some data to write */
+ if (count == 0) {
+ dbg(1, " %s : write request of 0 bytes", __FUNCTION__);
+ goto exit;
+ }
+
+ while (count > 0) {
+ if (dev->interrupt_out_urb->status == -EINPROGRESS) {
+ timeout = COMMAND_TIMEOUT;
+
+ while (timeout > 0) {
+ if (signal_pending(current)) {
+ dbg(1, " %s : interrupted", __FUNCTION__);
+ retval = -EINTR;
+ goto exit;
+ }
+ up(&dev->sem);
+ timeout = interruptible_sleep_on_timeout(&dev->write_wait, timeout);
+ down_interruptible(&dev->sem);
+ if (timeout > 0) {
+ break;

```


Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ bytes_written += bytes_to_write;
+ }
+ }
+
+ retval = bytes_written;
+
+exit:
+ /* unlock the device */
+ up(&dev->sem);
+
+ dbg(2, " %s : leave, return value %d", __FUNCTION__, retval);
+
+ return retval;
+}
+
+/**
+ * adu_probe
+ *
+ * Called by the usb core when a new device is connected that it thinks
+ * this driver might be interested in.
+ */
+static void *adu_probe(struct usb_device *udev,
+ unsigned int ifnum, const struct usb_device_id *id)
+{
+ struct adu_device *dev = NULL;
+ struct usb_interface *interface;
+ struct usb_interface_descriptor *iface_desc;
+ struct usb_endpoint_descriptor *endpoint;
+ int minor;
+ int in_end_size;
+ int out_end_size;
+ int i;
+ char name[10];
+ void *retval = NULL;
+
+ dbg(2, " %s : enter", __FUNCTION__);
+
+ if (udev == NULL) {
+ info ("udev is NULL.");
+ goto exit;
+ }
+
+ if (ifnum != 0) {
+ info ("Strange interface number %d.", ifnum);
+ goto exit;
+ }
+
+ /* select a "subminor" number (part of a minor number) */
+ down (&minor_table_mutex);
+ for (minor = 0; minor < MAX_DEVICES; ++minor) {
+ if (minor_table[minor] == NULL)
```

```
+ break;
+ }
+ if (minor >= MAX_DEVICES) {
+ info ("Too many devices plugged in, can not handle this device.");
+ goto unlock_minor_exit;
+ }
+
+ /* allocate memory for our device state and initialize it */
+ dev = kmalloc(sizeof(struct adu_device), GFP_KERNEL);
+ if (dev == NULL) {
+ err ("Out of memory");
+ goto unlock_minor_exit;
+ }
+ memset (dev, 0x00, sizeof (*dev));
+
+ init_MUTEX(&dev->sem);
+ spin_lock_init(&dev->buflock);
+ down(&dev->sem);
+ dev->udev = udev;
+
+ interface = &dev->udev->actconfig->interface[0];
+ iface_desc = &interface->altsetting[0];
+
+ //dev->interface = interface;
+ dev->minor = minor;
+ dev->open_count = 0;
+
+ dev->read_buffer_primary = NULL;
+ dev->read_buffer_secondary = NULL;
+ dev->read_buffer_length = 0;
+
+ dev->secondary_head = 0;
+ dev->secondary_tail = 0;
+
+ init_waitqueue_head(&dev->read_wait);
+ init_waitqueue_head(&dev->write_wait);
+
+ dev->interrupt_in_buffer = NULL;
+ dev->interrupt_in_endpoint = NULL;
+ dev->interrupt_in_urb = NULL;
+
+ dev->interrupt_out_buffer = NULL;
+ dev->interrupt_out_endpoint = NULL;
+ dev->interrupt_out_urb = NULL;
+
+ dev->read_urb_finished = 0;
+
+ /* set up the endpoint information */
+ for (i = 0; i < iface_desc->bNumEndpoints; ++i) {
+ endpoint = &iface_desc->endpoint[i];
+ }
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ /**
+ * Check if the endpoint has IN direction
+ * and check if the endpoint has interrupt transfer type
+ */
+ if (((endpoint->bEndpointAddress & USB_ENDPOINT_DIR_MASK) == USB_DIR_IN) &&
+ ((endpoint->bmAttributes & USB_ENDPOINT_XFERTYPE_MASK) ==
USB_ENDPOINT_XFER_INT)) {
+ /* we found an interrupt in endpoint */
+ dev->interrupt_in_endpoint = endpoint;
+ }
+
+ /**
+ * Check if the endpoint has OUT direction
+ * and check if the endpoint has interrupt transfer type
+ */
+ if (((endpoint->bEndpointAddress & USB_ENDPOINT_DIR_MASK) == USB_DIR_OUT) &&
+ ((endpoint->bmAttributes & USB_ENDPOINT_XFERTYPE_MASK) ==
USB_ENDPOINT_XFER_INT)) {
+ /* we found an interrupt out endpoint */
+ dev->interrupt_out_endpoint = endpoint;
+ }
+ }
+
+ if(dev->interrupt_in_endpoint == NULL) {
+ err("interrupt in endpoint not found");
+ goto error;
+ }
+ if (dev->interrupt_out_endpoint == NULL) {
+ err("interrupt out endpoint not found");
+ goto error;
+ }
+
+ in_end_size = le16_to_cpu(dev->interrupt_in_endpoint->wMaxPacketSize);
+ out_end_size = le16_to_cpu(dev->interrupt_out_endpoint->wMaxPacketSize);
+
+ dev->read_buffer_primary = kmalloc((4 * in_end_size), GFP_KERNEL);
+ if (!dev->read_buffer_primary) {
+ err("Couldn't allocate read_buffer_primary");
+ goto error;
+ }
+
+ /* debug code prime the buffer */
+ memset(dev->read_buffer_primary, 'a', in_end_size);
+ memset(dev->read_buffer_primary + in_end_size, 'b', in_end_size);
+ memset(dev->read_buffer_primary + (2 * in_end_size), 'c', in_end_size);
+ memset(dev->read_buffer_primary + (3 * in_end_size), 'd', in_end_size);
+
+ dev->read_buffer_secondary = kmalloc((4 * in_end_size), GFP_KERNEL);
+ if (!dev->read_buffer_secondary) {
+ err("Couldn't allocate read_buffer_secondary");
+ goto error;
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ }
+
+ /* debug code prime the buffer */
+ memset(dev->read_buffer_secondary, 'e', in_end_size);
+ memset(dev->read_buffer_secondary + in_end_size, 'f', in_end_size);
+ memset(dev->read_buffer_secondary + (2 * in_end_size), 'g', in_end_size);
+ memset(dev->read_buffer_secondary + (3 * in_end_size), 'h', in_end_size);
+
+ dev->interrupt_in_buffer = kmalloc(in_end_size, GFP_KERNEL);
+ if (!dev->interrupt_in_buffer) {
+ err("Couldn't allocate interrupt_in_buffer");
+ goto error;
+ }
+
+ /* debug code prime the buffer */
+ memset(dev->interrupt_in_buffer, 'i', in_end_size);
+
+ dev->interrupt_in_urb = usb_alloc_urb(0);
+ if (!dev->interrupt_in_urb) {
+ err("Couldn't allocate interrupt_in_urb");
+ goto error;
+ }
+
+ dev->interrupt_out_buffer = kmalloc(out_end_size, GFP_KERNEL);
+ if (!dev->interrupt_out_buffer) {
+ err("Couldn't allocate interrupt_out_buffer");
+ goto error;
+ }
+
+ dev->interrupt_out_urb = usb_alloc_urb(0);
+ if (!dev->interrupt_out_urb) {
+ err("Couldn't allocate interrupt_out_urb");
+ goto error;
+ }
+
+ /* put dev in interface->private_data for disconnect */
+ udev->actconfig->interface[0].private_data = dev;
+
+ /* publish it */
+ minor_table[minor] = dev;
+
+ /* initialize the devfs node for this device and register it */
+ sprintf(name, "adutux%d", dev->minor);
+
+ dev->devfs = devfs_register(usb_devfs_handle, name,
+ DEVFS_FL_DEFAULT, USB_MAJOR,
+ ADU_MINOR_BASE + dev->minor,
+ S_IFCHR | S_IRUSR | S_IWUSR |
+ S_IRGRP | S_IWGRP | S_IROTH,
+ &adu_fops, NULL);
+
+ /* let the user know what node this device is now attached to */
+ info ("ADU%d now attached to /dev/usb/adutux%d", udev->descriptor.idProduct, dev->minor);
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+
+ retval = dev;
+ /* unlock device */
+ up(&dev->sem);
+ goto unlock_minor_exit;
+
+error:
+ /* unlock device */
+ up(&dev->sem);
+ adu_delete(dev);
+ dev = NULL;
+
+unlock_minor_exit:
+ up(&minor_table_mutex);
+
+exit:
+ dbg(2, " %s : leave, return value 0x%.8lx (dev)", __FUNCTION__, (dev)?(long)dev:0);
+
+ return retval;
+ }
+
+/**
+ * adu_disconnect
+ *
+ * Called by the usb core when the device is removed from the system.
+ */
+static void adu_disconnect(struct usb_device *udev, void *ptr)
+{
+ struct adu_device *dev;
+ int minor;
+
+ dbg(2, " %s : enter", __FUNCTION__);
+
+ dev = (struct adu_device *)ptr;
+
+ down (&minor_table_mutex);
+ down (&dev->sem);
+
+ minor = dev->minor;
+
+ /* remove our devfs node */
+ devfs_unregister(dev->devfs);
+
+ /* if the device is not opened, then we clean up right now */
+ dbg(2, " %s : open count %d", __FUNCTION__, dev->open_count);
+ if (!dev->open_count) {
+ up (&dev->sem);
+ adu_delete(dev);
+ } else {
+ dev->udev = NULL;
+ up (&dev->sem);
+ }
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
+ }
+
+ up (&minor_table_mutex);
+ info("ADU device adutux%d now disconnected", minor);
+
+ dbg(2, " %s : leave", __FUNCTION__);
+}
+
+/**
+ * adu_init
+ */
+static int __init adu_init(void)
+{
+ int result;
+
+ dbg(2, " %s : enter", __FUNCTION__);
+
+ /* register this driver with the USB subsystem */
+ result = usb_register(&adu_driver);
+ if (result < 0) {
+ err("usb_register failed for the "__FILE__" driver. "
+ "Error number %d", result);
+ goto exit;
+ }
+
+ info("adutux " DRIVER_DESC " " DRIVER_VERSION);
+ info("adutux is an experimental driver. Use at your own risk");
+
+exit:
+ dbg(2, " %s : leave, return value %d", __FUNCTION__, result);
+
+ return result;
+}
+
+/**
+ * adu_exit
+ */
+static void __exit adu_exit(void)
+{
+ dbg(2, " %s : enter", __FUNCTION__);
+ /* deregister this driver with the USB subsystem */
+ usb_deregister(&adu_driver);
+ dbg(2, " %s : leave", __FUNCTION__);
+}
+
+module_init(adu_init);
+module_exit(adu_exit);
+
+MODULE_AUTHOR(DRIVER_AUTHOR);
+MODULE_DESCRIPTION(DRIVER_DESC);
+MODULE_LICENSE("GPL");
```

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.

```
diff -uprN -X dontdiff KERNELS/linux-2.4.35.3/drivers/usb/Config.in
linux-2.4.35.3.build/drivers/usb/Config.in
--- KERNELS/linux-2.4.35.3/drivers/usb/Config.in 2007-09-24 01:02:58.000000000 +0300
+++ linux-2.4.35.3.build/drivers/usb/Config.in 2007-10-10 19:49:19.000000000 +0300
@@ -102,6 +102,7 @@ if [ "$CONFIG_USB" = "y" -o "$CONFIG_US
comment 'USB Miscellaneous drivers'
dep_tristate ' USB Diamond Rio500 support (EXPERIMENTAL)' CONFIG_USB_RIO500 $CONFIG_USB
$CONFIG_EXPERIMENTAL
+ dep_tristate ' ADU devices from Ontrak Control Systems (EXPERIMENTAL)' CONFIG_USB_ADUTUX
$CONFIG_USB $CONFIG_EXPERIMENTAL
dep_tristate ' Auerswald device support' CONFIG_USB_AUERSWALD $CONFIG_USB
dep_tristate ' Texas Instruments Graph Link USB (aka SilverLink) cable support' CONFIG_USB_TIGL
$CONFIG_USB
dep_tristate ' Tieman Voyager USB Braille display support (EXPERIMENTAL)' CONFIG_USB_BRLVGER
$CONFIG_USB $CONFIG_EXPERIMENTAL
diff -uprN -X dontdiff KERNELS/linux-2.4.35.3/drivers/usb/Makefile
linux-2.4.35.3.build/drivers/usb/Makefile
--- KERNELS/linux-2.4.35.3/drivers/usb/Makefile 2007-09-24 01:02:58.000000000 +0300
+++ linux-2.4.35.3.build/drivers/usb/Makefile 2007-10-10 19:49:19.000000000 +0300
@@ -112,6 +112,7 @@ obj-$(CONFIG_USB_CATC) += catc.o
obj-$(CONFIG_USB_KAWETH) += kaweth.o
obj-$(CONFIG_USB_CDCETHER) += CDCEther.o
obj-$(CONFIG_USB_RIO500) += rio500.o
+obj-$(CONFIG_USB_ADUTUX) += adutux.o
obj-$(CONFIG_USB_TIGL) += tiglusb.o
obj-$(CONFIG_USB_DSBR) += dsbr100.o
obj-$(CONFIG_USB_MICROTEK) += microtek.o
diff -uprN -X dontdiff KERNELS/linux-2.4.35.3/MAINTAINERS linux-2.4.35.3.build/MAINTAINERS
--- KERNELS/linux-2.4.35.3/MAINTAINERS 2007-09-24 01:02:58.000000000 +0300
+++ linux-2.4.35.3.build/MAINTAINERS 2007-10-16 16:31:14.000000000 +0300
@@ -173,6 +173,11 @@ M: Thorsten Knabe <linux@thorsten-knabe.
W: http://linux.thorsten-knabe.de
S: Maintained
```

+ADUTUX ONTRAK CONTROL SYSTEMS USB DRIVER

+P: Vitaliy Ivanov

+M: vitalivanov@xxxxxxxxxx

+S: Maintained

+

ADVANSYS SCSI DRIVER

P: Bob Frey

M: linux@xxxxxxxxxxxxxx

-

To unsubscribe from this list: send the line "unsubscribe linux-kernel" in the body of a message to majordomo@xxxxxxxxxxxxxxxxxx

More majordomo info at <http://vger.kernel.org/majordomo-info.html>

Please read the FAQ at <http://www.tux.org/lkml/>

Re: [2.4 patch] Port of adutux driver from 2.6 kernel to 2.4.