

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2008-06/msg11348.html>

- *From:* Henrik Rydberg <rydberg@xxxxxxxxxxx>
 - *Date:* Sat, 28 Jun 2008 00:17:50 +0200
-

On Fri, 2008-06-27 at 09:43 +0200, Oliver Neukum wrote:

Am Freitag 27 Juni 2008 01:07:19 schrieb Henrik Rydberg:

From: Henrik Rydberg <rydberg@xxxxxxxxxxx>

BCM5974: This driver adds support for the multitouch trackpad on the new Apple Macbook Air and Macbook Pro Penryn laptops. It replaces the appletouch driver on those computers, and integrates well with the synaptics driver of the Xorg system.

Some comments on the driver.

Regards
Oliver

Thank you very much for finding the bugs, and for your general comments. I have modified the latest version accordingly, taking all of your suggestions into account. The new patch follows below.

```
---
diff --git a/drivers/input/mouse/Kconfig b/drivers/input/mouse/Kconfig
index 7bbea09..89ef7c3 100644
--- a/drivers/input/mouse/Kconfig
+++ b/drivers/input/mouse/Kconfig
@@ -130,6 +130,29 @@ config MOUSE_APPLETOUCH
To compile this driver as a module, choose M here: the
module will be called appletouch.
```

```
+config MOUSE_BCM5974
+ tristate "Apple USB BCM5974 Multitouch trackpad support"
+ depends on USB_ARCH_HAS_HCD
+ select USB
+ help
+ Say Y here if you have an Apple USB BCM5974 Multitouch
```

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

```
+ trackpad.
+
+ The BCM5974 is the multitouch trackpad found in the Macbook
+ Air (JAN2008) and Macbook Pro Penryn (FEB2008) laptops.
+
+ It is also found in the iPhone (2007) and Ipod Touch (2008).
+
+ This driver provides multitouch functionality together with
+ the synaptics X11 driver.
+
+ The interface is currently identical to the appletouch interface,
+ for further information, see
+ <file:Documentation/input/appletouch.txt>.
+
+ To compile this driver as a module, choose M here: the
+ module will be called bcm5974.
+
config MOUSE_INPORT
tristate "InPort/MS/ATIXL busmouse"
depends on ISA
diff --git a/drivers/input/mouse/Makefile b/drivers/input/mouse/Makefile
index 9e6e363..d4d2025 100644
--- a/drivers/input/mouse/Makefile
+++ b/drivers/input/mouse/Makefile
@@ -6,6 +6,7 @@

obj-$(CONFIG_MOUSE_AMIGA) += amimouse.o
obj-$(CONFIG_MOUSE_APPLETOUCH) += appletouch.o
+obj-$(CONFIG_MOUSE_BCM5974) += bcm5974.o
obj-$(CONFIG_MOUSE_ATARI) += atarimouse.o
obj-$(CONFIG_MOUSE_RISCPC) += rpcmouse.o
obj-$(CONFIG_MOUSE_INPORT) += inport.o
diff -uprN -X upstream-2.6/Documentation/dontdiff baseline-2.6/drivers/input/mouse/bcm5974.c
upstream-2.6/drivers/input/mouse/bcm5974.c
--- baseline-2.6/drivers/input/mouse/bcm5974.c 1970-01-01 01:00:00.000000000 +0100
+++ upstream-2.6/drivers/input/mouse/bcm5974.c 2008-06-27 23:59:42.000000000 +0200
@@ -0,0 +1,667 @@
+/*
+ * Apple USB BCM5974 (Macbook Air and Penryn Macbook Pro) multitouch driver
+ *
+ * Copyright (C) 2008 Henrik Rydberg (rydberg@xxxxxxxxxxxx)
+ *
+ * The USB initialization and package decoding was made by
+ * Scott Shawcroft as part of the touchd user-space driver project:
+ * Copyright (C) 2008 Scott Shawcroft (tannewt of tannewt.org)
+ *
+ * The BCM5974 driver is based on the appletouch driver:
+ * Copyright (C) 2001-2004 Greg Kroah-Hartman (greg@xxxxxxxx)
+ * Copyright (C) 2005 Johannes Berg (johannes@xxxxxxxxxxxxxxxx)
+ * Copyright (C) 2005 Stelian Pop (stelian@xxxxxxxxxxxx)
+ * Copyright (C) 2005 Frank Arnold (frank@xxxxxxxxxxxxxxxxxxxxxxxx)
```

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

```
+ * Copyright (C) 2005 Peter Osterlund (petero2@xxxxxxxxxx)
+ * Copyright (C) 2005 Michael Hanselmann (linux-kernel@xxxxxxxxxx)
+ * Copyright (C) 2006 Nicolas Boichat (nicolas@xxxxxxxxxx)
+ *
+ * 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.
+ *
+ * This program is distributed in the hope that it will be useful,
+ * but WITHOUT ANY WARRANTY; without even the implied warranty of
+ * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
+ * GNU General Public License for more details.
+ *
+ * You should have received a copy of the GNU General Public License
+ * along with this program; if not, write to the Free Software
+ * Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.
+ *
+ */
+
+#include <linux/kernel.h>
+#include <linux/errno.h>
+#include <linux/init.h>
+#include <linux/slab.h>
+#include <linux/module.h>
+#include <linux/usb/input.h>
+
+#define APPLE_VENDOR_ID 0x05AC
+
+/* MacbookAir BCM5974, aka wellspring */
+
+#define ATP_WELLSPRING_ANSI 0x0223
+#define ATP_WELLSPRING_ISO 0x0224
+#define ATP_WELLSPRING_JIS 0x0225
+#define ATP_WELLSPRING2_ANSI 0x0230
+#define ATP_WELLSPRING2_ISO 0x0231
+#define ATP_WELLSPRING2_JIS 0x0232
+
+#define ATP_DEVICE(prod) { \
+ .match_flags = (USB_DEVICE_ID_MATCH_DEVICE | \
+ USB_DEVICE_ID_MATCH_INT_CLASS | \
+ USB_DEVICE_ID_MATCH_INT_PROTOCOL), \
+ .idVendor = APPLE_VENDOR_ID, \
+ .idProduct = (prod), \
+ .bInterfaceClass = 0x03, \
+ .bInterfaceProtocol = 0x02 \
+ }
+
+/* table of devices that work with this driver */
+static const struct usb_device_id atp_table [] = {
+ /* MacbookAir1.1 */
```

```
+ ATP_DEVICE(ATP_WELLSPRING_ANSI),
+ ATP_DEVICE(ATP_WELLSPRING_ISO),
+ ATP_DEVICE(ATP_WELLSPRING_JIS),
+
+ /* MacbookProPenryn */
+ ATP_DEVICE(ATP_WELLSPRING2_ANSI),
+ ATP_DEVICE(ATP_WELLSPRING2_ISO),
+ ATP_DEVICE(ATP_WELLSPRING2_JIS),
+
+ /* Terminating entry */
+ {}
+};
+MODULE_DEVICE_TABLE(usb, atp_table);
+
+struct atp_params_t {
+ int devmin; /* device minimum reading */
+ int devmax; /* device maximum reading */
+ int min; /* logical minimum reading */
+ int max; /* logical maximum reading */
+ int fuzz; /* reading noise value */
+ int flat; /* zero */
+};
+
+struct atp_config_t {
+ int ansi, iso, jis; /* the product id of this device */
+ int bt_ep; /* the endpoint of the button interface */
+ int bt_datalen; /* data length of the button interface */
+ int tp_ep; /* the endpoint of the trackpad interface */
+ int tp_datalen; /* data length of the trackpad interface */
+ struct atp_params_t x; /* horizontal limits */
+ struct atp_params_t y; /* vertical limits */
+ struct atp_params_t p; /* pressure limits */
+};
+
+/* trackpad header structure */
+struct tp_header_t {
+ u8 unknown1[16]; /* constants, timers, etc */
+ u8 nfinger; /* number of fingers on trackpad */
+ u8 unknown2[9]; /* constants, timers, etc */
+};
+
+/* trackpad finger structure */
+struct tp_finger_t {
+ __le16 origin; /* left/right origin? */
+ __le16 abs_x; /* absolute x coordinate */
+ __le16 abs_y; /* absolute y coordinate */
+ __le16 rel_x; /* relative x coordinate */
+ __le16 rel_y; /* relative y coordinate */
+ __le16 size_major; /* finger size, major axis? */
+ __le16 size_minor; /* finger size, minor axis? */
+ __le16 orientation; /* 16384 when point, else 15 bit angle */
+};
```

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

```
+__le16 force_major; /* trackpad force, major axis? */
+__le16 force_minor; /* trackpad force, minor axis? */
+__le16 unused[3]; /* zeros */
+__le16 multi; /* one finger: varies, more fingers: constant */
+};
+
+/* trackpad data structure */
+struct tp_data_t {
+ struct tp_header_t header;
+ struct tp_finger_t finger[16];
+};
+
+/* device constants */
+static const struct atp_config_t atp_config_table[] = {
+ {
+ ATP_WELLSPRING_ANSI,
+ ATP_WELLSPRING_ISO,
+ ATP_WELLSPRING_JIS,
+ 0x84, 4,
+ 0x81, sizeof(struct tp_data_t),
+ {-4824, 5342, 0, 1280, 16, 0},
+ {-172, 5820, 0, 800, 16, 0},
+ {0, 256, 0, 256, 16, 0}
+ },
+ {
+ ATP_WELLSPRING2_ANSI,
+ ATP_WELLSPRING2_ISO,
+ ATP_WELLSPRING2_JIS,
+ 0x84, 4,
+ 0x81, sizeof(struct tp_data_t),
+ {-4824, 5342, 0, 1280, 16, 0},
+ {-172, 5820, 0, 800, 16, 0},
+ {0, 256, 0, 256, 16, 0}
+ },
+ {}
+};
+
+static inline
+const struct atp_config_t *atp_product_config(struct usb_device *udev)
+{
+ u16 id = le16_to_cpu(udev->descriptor.idProduct);
+ const struct atp_config_t *config;
+ for (config = atp_config_table; config->ansi; ++config)
+ if (config->ansi == id || config->iso == id || config->jis == id)
+ return config;
+ return atp_config_table;
+}
+
+/* Wellspring initialization constants */
+#define ATP_WELLSPRING_MODE_READ_REQUEST_ID 1
+#define ATP_WELLSPRING_MODE_WRITE_REQUEST_ID 9
```

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

```
+ #define ATP_WELLSPRING_MODE_REQUEST_VALUE 0x300
+ #define ATP_WELLSPRING_MODE_REQUEST_INDEX 0
+ #define ATP_WELLSPRING_MODE_VENDOR_VALUE_1 0x01
+ #define ATP_WELLSPRING_MODE_VENDOR_VALUE_2 0x05
+
+ #define dprintk(format, a...) \
+ { if (debug) printk(KERN_DEBUG format, ##a); }
+
+ MODULE_AUTHOR("Henrik Rydberg");
+ MODULE_DESCRIPTION("Apple USB BCM5974 multitouch driver");
+ MODULE_LICENSE("GPL");
+
+ static int debug = 1;
+ module_param(debug, int, 0644);
+ MODULE_PARM_DESC(debug, "Activate debugging output");
+
+ static int atp_wellspring_init(struct usb_device *udev)
+ {
+     const struct atp_config_t *cfg = atp_product_config(udev);
+     char *data = kmalloc(8, GFP_DMA);
+     int size;
+
+     /* reset button endpoint */
+     if (usb_control_msg(udev, usb_rcvctrlpipe(udev), 0),
+         USB_REQ_CLEAR_FEATURE, USB_RECIP_ENDPOINT,
+         0, cfg->bt_ep, NULL, 0, 5000)) {
+         err("Could not reset button endpoint");
+         goto error;
+     }
+
+     /* reset trackpad endpoint */
+     if (usb_control_msg(udev, usb_rcvctrlpipe(udev), 0),
+         USB_REQ_CLEAR_FEATURE, USB_RECIP_ENDPOINT,
+         0, cfg->tp_ep, NULL, 0, 5000)) {
+         err("Could not reset trackpad endpoint");
+         goto error;
+     }
+
+     /* read configuration */
+     size = usb_control_msg(udev, usb_rcvctrlpipe(udev), 0),
+         ATP_WELLSPRING_MODE_READ_REQUEST_ID,
+         USB_DIR_IN | USB_TYPE_CLASS | USB_RECIP_INTERFACE,
+         ATP_WELLSPRING_MODE_REQUEST_VALUE,
+         ATP_WELLSPRING_MODE_REQUEST_INDEX, data, 8, 5000);
+
+     if (size != 8) {
+         err("Could not do mode read request from device (Wellspring Raw mode)");
+         goto error;
+     }
+
+     /* apply the mode switch */
```

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

```
+ data[0] = ATP_WELLSPRING_MODE_VENDOR_VALUE_1;
+ data[1] = ATP_WELLSPRING_MODE_VENDOR_VALUE_2;
+
+ /* write configuration */
+ size = usb_control_msg(udev, usb_sndctrlpipe(udev, 0),
+ ATP_WELLSPRING_MODE_WRITE_REQUEST_ID,
+ USB_DIR_OUT | USB_TYPE_CLASS | USB_RECIP_INTERFACE,
+ ATP_WELLSPRING_MODE_REQUEST_VALUE,
+ ATP_WELLSPRING_MODE_REQUEST_INDEX, data, 8, 5000);
+
+ if (size != 8) {
+ err("Could not do mode write request to device (Wellspring Raw mode)");
+ goto error;
+ }
+
+ kfree(data);
+ return 0;
+
+ error:
+ kfree(data);
+ return -EIO;
+}
+
+ /* Structure to hold all of our device specific stuff */
+ struct atp {
+ char phys[64];
+ struct usb_device *udev; /* usb device */
+ struct input_dev *input; /* input dev */
+ struct atp_config_t cfg; /* device configuration */
+ unsigned open; /* non-zero if opened */
+ struct urb *bt_urb; /* button usb request block */
+ signed char *bt_data; /* button transferred data */
+ unsigned bt_valid; /* are the button sensors valid ? */
+ unsigned bt_state; /* current button state */
+ struct urb *tp_urb; /* trackpad usb request block */
+ signed char *tp_data; /* trackpad transferred data */
+ unsigned tp_valid; /* are the trackpad sensors valid ? */
+};
+
+ static inline int raw2int(__le16 x)
+ {
+ return (short)le16_to_cpu(x);
+ }
+
+ static inline int int2scale(const struct atp_params_t *p, int x)
+ {
+ return ((p->max-p->min)*x)/(p->devmax-p->devmin);
+ }
+
+ /**
+ * all value ranges, both device and logical, are assumed to be [a,b).
```

```

+ */
+static inline int int2bound(const struct atp_params_t *p, int x)
+{
+ int s = p->min+int2scale(p, x);
+ return s < p->min?p->min:s >= p->max?p->max-1:s;
+}
+
+/**
+ * check quality of reading.
+ * -1: bad data
+ * 0: ignore this reading
+ * 1: good reading
+ */
+static int compute_quality(const unsigned char *data, int size)
+{
+ if (size < 26 || (size-26)%28 != 0)
+ return -1;
+
+ return 1;
+}
+
+/**
+ * convert raw data to synaptics sensor output.
+ * returns the number of fingers on the trackpad,
+ * or a negative number in vase of bad data.
+ */
+static int compute_movement(int *abs_x, int *abs_y,
+ int *rel_x, int *rel_y,
+ int *pressure,
+ struct atp_config_t *cfg,
+ const unsigned char *data, int size)
+{
+ const int nfinger = (size-26)/28;
+ const struct tp_data_t *tp = (struct tp_data_t *) data;
+
+ if (nfinger) {
+ *abs_x = int2bound(&cfg->x, raw2int(tp->finger->abs_x) - cfg->x.devmin);
+ *abs_y = int2bound(&cfg->y, cfg->y.devmax - raw2int(tp->finger->abs_y));
+ *rel_x = int2scale(&cfg->x, raw2int(tp->finger->rel_x));
+ *rel_y = int2scale(&cfg->y, -raw2int(tp->finger->rel_y));
+ *pressure = int2bound(&cfg->p, raw2int(tp->finger->force_major));
+ } else {
+ *abs_x = 0;
+ *abs_y = 0;
+ *rel_x = 0;
+ *rel_y = 0;
+ *pressure = 0;
+ }
+
+ /* report zero fingers for zero pressure */
+ return *pressure > 0?nfinger:0;

```

```

+}
+
+static void atp_button(struct urb *urb)
+{
+ struct atp *dev = urb->context;
+ const unsigned char *data = dev->bt_data;
+ const int size = dev->bt_urb->actual_length;
+ int button = 0, retval;
+
+ switch (urb->status) {
+ case 0:
+ /* success */
+ break;
+ case -EOVERFLOW:
+ case -ECONNRESET:
+ case -ENOENT:
+ case -ESHUTDOWN:
+ /* This urb is terminated, clean up */
+ dbg("%s - urb shutting down with status: %d",
+ __func__, urb->status);
+ return;
+ default:
+ dbg("%s - nonzero urb status received: %d",
+ __func__, urb->status);
+ goto exit;
+ }
+
+ /* first sample data ignored */
+ if (!dev->bt_valid) {
+ dev->bt_valid = 1;
+ goto exit;
+ }
+
+ /* drop incomplete datasets */
+ if (size != 4) {
+ dprintk("bcm5974: incomplete button package (first byte: %d, length: %d)\n",
+ (int)data[0], size);
+ goto exit;
+ }
+
+ button = data[1];
+
+ /* only report button state changes */
+ if (button != dev->bt_state) {
+ input_report_key(dev->input, BTN_LEFT, button);
+ input_sync(dev->input);
+ }
+
+ dev->bt_state = button;
+
+ exit:

```

```

+ retval = usb_submit_urb(dev->bt_urb, GFP_ATOMIC);
+ if (retval) {
+ err("%s - button usb_submit_urb failed with result %d",
+ __func__, retval);
+ }
+ }
+
+static void atp_trackpad(struct urb *urb)
+{
+ struct atp *dev = urb->context;
+ int abs_x, abs_y, rel_x, rel_y, pressure;
+ int quality, nfinger, retval;
+
+ switch (urb->status) {
+ case 0:
+ /* success */
+ break;
+ case -EOVERFLOW:
+ case -ECONNRESET:
+ case -ENOENT:
+ case -ESHUTDOWN:
+ /* This urb is terminated, clean up */
+ dbg("%s - urb shutting down with status: %d",
+ __func__, urb->status);
+ return;
+ default:
+ dbg("%s - nonzero urb status received: %d",
+ __func__, urb->status);
+ goto exit;
+ }
+
+ /* first sample data ignored */
+ if (!dev->tp_valid) {
+ dev->tp_valid = 1;
+ goto exit;
+ }
+
+ /* determine quality of reading */
+ quality = compute_quality(dev->tp_data, dev->tp_urb->actual_length);
+
+ /* drop incomplete datasets */
+ if (quality < 0) {
+ dprintk("bcm5974: incomplete trackpad package "
+ "(first byte: %d, length: %d)\n",
+ (int)dev->tp_data[0], dev->tp_urb->actual_length);
+ goto exit;
+ }
+
+ /* drop poor quality readings */
+ if (quality == 0)
+ goto exit;

```

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

```
+
+ nfinger = compute_movement(&abs_x, &abs_y, &rel_x, &rel_y, &pressure,
+ &dev->cfg,
+ dev->tp_data, dev->tp_urb->actual_length);
+
+ if (debug > 1) {
+ printk(KERN_DEBUG "bcm5974: x: %04d y: %04d dx: %3d dy: %3d p: %3d\n",
+ abs_x, abs_y, rel_x, rel_y, pressure);
+ }
+
+ /* input_report_key(dev->input, BTN_TOUCH, pressure > dev->cfg.p.fuzz); */
+ input_report_abs(dev->input, ABS_PRESSURE, pressure);
+ input_report_abs(dev->input, ABS_X, abs_x);
+ input_report_abs(dev->input, ABS_Y, abs_y);
+ /* input_report_rel(dev->input, REL_X, rel_x); */
+ /* input_report_rel(dev->input, REL_Y, rel_y); */
+ input_report_key(dev->input, BTN_TOOL_FINGER, nfinger == 1);
+ input_report_key(dev->input, BTN_TOOL_DOUBLETAP, nfinger == 2);
+ input_report_key(dev->input, BTN_TOOL_TRIPLETAP, nfinger > 2);
+ input_sync(dev->input);
+
+ exit:
+ retval = usb_submit_urb(dev->tp_urb, GFP_ATOMIC);
+ if (retval) {
+ err("%s - trackpad usb_submit_urb failed with result %d",
+ __func__, retval);
+ }
+ }
+
+static int atp_open(struct input_dev *input)
+{
+ struct atp *dev = input_get_drvdata(input);
+
+ if (usb_submit_urb(dev->bt_urb, GFP_ATOMIC))
+ goto error;
+
+ if (usb_submit_urb(dev->tp_urb, GFP_ATOMIC))
+ goto err_free_bt_urb;
+
+ dev->open = 1;
+ return 0;
+
+ err_free_bt_urb:
+ usb_free_urb(dev->bt_urb);
+ error:
+ return -EIO;
+ }
+
+static void atp_close(struct input_dev *input)
+{
+ struct atp *dev = input_get_drvdata(input);
```

```

+
+ usb_kill_urb(dev->tp_urb);
+ usb_kill_urb(dev->bt_urb);
+ dev->open = 0;
+ }
+
+ static int atp_probe(struct usb_interface *iface,
+ const struct usb_device_id *id)
+ {
+ int error = -ENOMEM;
+ struct usb_device *udev = interface_to_usbdev(iface);
+ const struct atp_config_t *cfg;
+ struct atp *dev;
+ struct input_dev *input_dev;
+
+ /* find the product index */
+ cfg = atp_product_config(udev);
+
+ /* allocate memory for our device state and initialize it */
+ dev = kzalloc(sizeof(struct atp), GFP_KERNEL);
+ input_dev = input_allocate_device();
+ if (!dev || !input_dev) {
+ err("Out of memory");
+ goto err_free_devs;
+ }
+
+ dev->udev = udev;
+ dev->input = input_dev;
+ dev->cfg = *cfg;
+
+ /* switch to raw sensor mode */
+ if (atp_wellspring_init(udev))
+ goto err_free_devs;
+
+ printk(KERN_INFO "bcm5974: Wellspring mode initialized.\n");
+
+ dev->bt_urb = usb_alloc_urb(0, GFP_KERNEL);
+ if (!dev->bt_urb)
+ goto err_free_devs;
+
+ dev->tp_urb = usb_alloc_urb(0, GFP_KERNEL);
+ if (!dev->tp_urb)
+ goto err_free_bt_urb;
+
+ dev->bt_data = usb_buffer_alloc(dev->udev,
+ dev->cfg.bt_data_len,
+ GFP_KERNEL,
+ &dev->bt_urb->transfer_dma);
+ if (!dev->bt_data)
+ goto err_free_urb;
+
+

```

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

```
+ dev->tp_data = usb_buffer_alloc(dev->udev,
+ dev->cfg.tp_dataalen,
+ GFP_KERNEL,
+ &dev->tp_urb->transfer_dma);
+ if (!dev->tp_data)
+ goto err_free_bt_buffer;
+
+ usb_fill_int_urb(dev->bt_urb, udev,
+ usb_rcvintpipe(udev, cfg->bt_ep),
+ dev->bt_data, dev->cfg.bt_dataalen,
+ atp_button, dev, 1);
+
+ usb_fill_int_urb(dev->tp_urb, udev,
+ usb_rcvintpipe(udev, cfg->tp_ep),
+ dev->tp_data, dev->cfg.tp_dataalen,
+ atp_trackpad, dev, 1);
+
+ usb_make_path(udev, dev->phys, sizeof(dev->phys));
+ strcat(dev->phys, "/input0", sizeof(dev->phys));
+
+ input_dev->name = "bcm5974";
+ input_dev->phys = dev->phys;
+ usb_to_input_id(dev->udev, &input_dev->id);
+ input_dev->dev.parent = &iface->dev;
+
+ input_set_drvdata(input_dev, dev);
+
+ input_dev->open = atp_open;
+ input_dev->close = atp_close;
+
+ set_bit(EV_ABS, input_dev->evbit);
+ input_set_abs_params(input_dev, ABS_X,
+ cfg->x.min, cfg->x.max, cfg->x.fuzz, cfg->x.flat);
+ input_set_abs_params(input_dev, ABS_Y,
+ cfg->y.min, cfg->y.max, cfg->y.fuzz, cfg->y.flat);
+ input_set_abs_params(input_dev, ABS_PRESSURE,
+ cfg->p.min, cfg->p.max, cfg->p.fuzz, cfg->p.flat);
+ /* set_bit(EV_REL, input_dev->evbit); */
+
+ set_bit(EV_KEY, input_dev->evbit);
+ /* set_bit(BTN_TOUCH, input_dev->keybit); */
+ set_bit(BTN_TOOL_FINGER, input_dev->keybit);
+ set_bit(BTN_TOOL_DOUBLETAP, input_dev->keybit);
+ set_bit(BTN_TOOL_TRIPLETAP, input_dev->keybit);
+ set_bit(BTN_LEFT, input_dev->keybit);
+
+ error = input_register_device(dev->input);
+ if (error)
+ goto err_free_buffer;
+
+ /* save our data pointer in this interface device */
```

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

```
+ usb_set_intfdata(iface, dev);
+
+ return 0;
+
+ err_free_buffer:
+ usb_buffer_free(dev->udev, dev->cfg.tp_datalen,
+ dev->tp_data, dev->tp_urb->transfer_dma);
+ err_free_bt_buffer:
+ usb_buffer_free(dev->udev, dev->cfg.bt_datalen,
+ dev->bt_data, dev->bt_urb->transfer_dma);
+ err_free_urb:
+ usb_free_urb(dev->tp_urb);
+ err_free_bt_urb:
+ usb_free_urb(dev->bt_urb);
+ err_free_devs:
+ usb_set_intfdata(iface, NULL);
+ kfree(dev);
+ input_free_device(input_dev);
+ return error;
+}
+
+static void atp_disconnect(struct usb_interface *iface)
+{
+ struct atp *dev = usb_get_intfdata(iface);
+
+ usb_set_intfdata(iface, NULL);
+ if (dev) {
+ input_unregister_device(dev->input);
+ usb_buffer_free(dev->udev, dev->cfg.tp_datalen,
+ dev->tp_data, dev->tp_urb->transfer_dma);
+ usb_buffer_free(dev->udev, dev->cfg.bt_datalen,
+ dev->bt_data, dev->bt_urb->transfer_dma);
+ usb_free_urb(dev->tp_urb);
+ usb_free_urb(dev->bt_urb);
+ kfree(dev);
+ }
+ printk(KERN_INFO "input: bcm5974 disconnected\n");
+}
+
+static int atp_suspend(struct usb_interface *iface, pm_message_t message)
+{
+ struct atp *dev = usb_get_intfdata(iface);
+
+ usb_kill_urb(dev->tp_urb);
+ dev->tp_valid = 0;
+
+ usb_kill_urb(dev->bt_urb);
+ dev->bt_valid = 0;
+
+ return 0;
+}
```

Re: [PATCH 001/001] linux-input: Support for BCM5974 multitouch trackpad

```
+
+static int atp_resume(struct usb_interface *iface)
+{
+ struct atp *dev = usb_get_intfdata(iface);
+
+ if (dev->open) {
+ if (usb_submit_urb(dev->bt_urb, GFP_ATOMIC))
+ goto error;
+ if (usb_submit_urb(dev->tp_urb, GFP_ATOMIC))
+ goto err_free_bt_urb;
+ }
+
+ return 0;
+
+ err_free_bt_urb:
+ usb_free_urb(dev->bt_urb);
+ error:
+ return -EIO;
+}
+
+static struct usb_driver atp_driver = {
+ .name = "bcm5974",
+ .probe = atp_probe,
+ .disconnect = atp_disconnect,
+ .suspend = atp_suspend,
+ .resume = atp_resume,
+ .id_table = atp_table,
+};
+
+static int __init atp_init(void)
+{
+ return usb_register(&atp_driver);
+}
+
+static void __exit atp_exit(void)
+{
+ usb_deregister(&atp_driver);
+}
+
+module_init(atp_init);
+module_exit(atp_exit);
```

--

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

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

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