

[7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2004-07/5391.html>

From: David Gibson (hermes_at_gibson.dropbear.id.au)

Date: 07/28/04

Date: Wed, 28 Jul 2004 16:56:59 +1000

To: Jeff Garzik <jgarzik@pobox.com>, Francois Romieu <romieu@fr.zoreil.com>, Linux kernel mailing

Various trivial cleanups to the orinoco driver: whitespace changes, spelling/capitalization errors corrected, some fairly insignificant comments added, removed or reformatted.

Signed-off-by: David Gibson <hermes@gibson.dropbear.id.au>

Index: working-2.6/drivers/net/wireless/orinoco.c

----- working-2.6.orig/drivers/net/wireless/orinoco.c 2004-07-28 14:57:52.310664496 +1000

+++ working-2.6/drivers/net/wireless/orinoco.c 2004-07-28 14:57:53.046552624 +1000

@@ -59,7 +59,7 @@

- * o Add PM timeout (holdover duration)
- * o Enable "iwconfig eth0 key off" and friends (toggle flags)
- * o Enable "iwconfig eth0 power unicast/all" (toggle flags)
- * o Try with an intel card. It report firmware 1.01, behave like
- + * o Try with an Intel card. It report firmware 1.01, behave like
- * an antiquated firmware, however on windows it says 2.00. Yuck !
- * o Workaround firmware bug in allocate buffer (Intel 1.01)
- * o Finish external renaming to orinoco...

@@ -69,8 +69,8 @@

- * o Update to Wireless 11 -> add retry limit/lifetime support
- * o Tested with a D-Link DWL 650 card, fill in firmware support
- * o Warning on Vcc mismatch (D-Link 3.3v card in Lucent 5v only slot)
- * o Fixed the Prims2 WEP bugs that I introduced in v0.03 :-(
- * It work on D-Link *only* after a tcpdump. Weird...
- + * o Fixed the Prism2 WEP bugs that I introduced in v0.03 :-(
- + * It works on D-Link *only* after a tcpdump. Weird...
- * And still doesn't work on Intel card. Grrrr...
- * o Update the mode after a setport3
- * o Add preamble setting for Symbol cards (not yet enabled)

@@ -106,7 +106,7 @@

- * o Remove deferred power enabling code
- *
- * v0.05c -> v0.05d - 5/5/2001 - Jean II
- * o Workaround to SNAP decapsulate frame from LinkSys AP

Linux-Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

- + * o Workaround to SNAP decapsulate frame from Linksys AP
 - * original patch from : Dong Liu <dliu AT research.bell-labs.com>
 - * (note : the memcmp bug was mine – fixed)
 - * o Remove set_retry stuff, no firmware support it (bloat--).
- @@ -135,15 +135,15 @@
 - *
 - * v0.06c -> v0.06d – 6/7/2001 – David Gibson
 - * o Change a bunch of KERN_INFO messages to KERN_DEBUG, as per Linus'
 - * wishes to reduce the number of unnecessary messages.
 - + * wishes to reduce the number of unnecessary messages.
 - * o Removed bogus message on CRC error.
 - * o Merged fixeds for v0.08 Prism 2 firmware from William Waghorn
 - + * o Merged fixes for v0.08 Prism 2 firmware from William Waghorn
 - * <willwaghorn AT yahoo.co.uk>
 - * o Slight cleanup/re-arrangement of firmware detection code.
 - *
 - * v0.06d -> v0.06e – 1/8/2001 – David Gibson
 - * o Removed some redundant global initializers (orinoco_cs.c).
 - * o Added some module metadataa
 - + * o Added some module metadata
 - *
 - * v0.06e -> v0.06f – 14/8/2001 – David Gibson
 - * o Wording fix to license
 - @@ -173,7 +173,7 @@
 - * o Turned has_big_wep on for Intersil cards. That's not true for all of them but we should at least let the capable ones try.
 - * o Wait for BUSY to clear at the beginning of hermes_bap_seek(). I
 - * realised that my assumption that the driver's serialization
 - + * realized that my assumption that the driver's serialization
 - * would prevent the BAP being busy on entry was possibly false, because
 - * things other than seeks may make the BAP busy.
 - * o Use "alternate" (oui 00:00:00) encapsulation by default.
 - @@ -182,12 +182,12 @@
 - * o Don't try to make __initdata const (the version string). This can't work because of the way the __initdata sectioning works.
 - * o Added MODULE_LICENSE tags.
 - * o Support for PLX (transparent PCMCIA->PCI brdge) cards.
 - * o Changed to using the new type-fascist min/max.
 - + * o Support for PLX (transparent PCMCIA->PCI bridge) cards.
 - + * o Changed to using the new type-fascist min/max.
 - *
 - * v0.08 -> v0.08a – 9/10/2001 – David Gibson
 - * o Inserted some missing acknowledgements/info into the Changelog.
 - * o Fixed some bugs in the normalisation of signal level reporting.
 - + * o Fixed some bugs in the normalization of signal level reporting.
 - * o Fixed bad bug in WEP key handling on Intersil and Symbol firmware, which led to an instant crash on big-endian machines.
 - *
 - @@ -343,7 +343,7 @@
 - * o Bugfix in orinoco_stop() – it used to fail if hw_unavailable was set, which was usually true on PCMCIA hot removes.

Linux–Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

```
* o Track LINKSTATUS messages, silently drop Tx packets before
- * we are connected (avoids cofusing the firmware), and only
+ * we are connected (avoids confusing the firmware), and only
* give LINKSTATUS printk()s if the status has changed.
*
* v0.13b -> v0.13c – 11 Mar 2003 – David Gibson
@@ -398,7 +398,8 @@
* o Disconnect wireless extensions from fundamental configuration.
* o (maybe) Software WEP support (patch from Stano Meduna).
* o (maybe) Use multiple Tx buffers – driver handling queue
- * rather than firmware. */
+ * rather than firmware.
+ */

/* Locking and synchronization:
*
@@ -415,7 +416,8 @@
* flag after taking the lock, and if it is set, give up on whatever
* they are doing and drop the lock again. The orinoco_lock()
* function handles this (it unlocks and returns –EBUSY if
- * hw_unavailable is non–zero). */
+ * hw_unavailable is non–zero).
+ */

#include <linux/config.h>

@@ -495,9 +497,10 @@
HERMES_MAX_MULTICAST : 0)*/
#define MAX_MULTICAST(priv) (HERMES_MAX_MULTICAST)

-#define ORINOCO_INTEN ( HERMES_EV_RX | HERMES_EV_ALLOC | HERMES_EV_TX | \
- HERMES_EV_TXEXC | HERMES_EV_WTERR | HERMES_EV_INFO | \
- HERMES_EV_INFDROP )
+#define ORINOCO_INTEN (HERMES_EV_RX | HERMES_EV_ALLOC \
+ | HERMES_EV_TX | HERMES_EV_TXEXC \
+ | HERMES_EV_WTERR | HERMES_EV_INFO \
+ | HERMES_EV_INFDROP )

/*****
/* Data tables */
@@ -510,7 +513,8 @@
};
#define NUM_CHANNELS ARRAY_SIZE(channel_frequency)

-/* This tables gives the actual meanings of the bitrate IDs returned by the firmware. */
+/* This tables gives the actual meanings of the bitrate IDs returned
+ * by the firmware. */
struct {
    int bitrate; /* in 100s of kilobits */
    int automatic;
@@ -574,8 +578,7 @@
```

```

/* Internal helper functions */
/*****

-static inline void
-set_port_type(struct orinoco_private *priv)
+static inline void set_port_type(struct orinoco_private *priv)
{
    switch (priv->iw_mode) {
    case IW_MODE_INFRA:
@@ -640,16 +643,14 @@
        return err;
    }
}

-struct net_device_stats *
-orinoco_get_stats(struct net_device *dev)
+struct net_device_stats *orinoco_get_stats(struct net_device *dev)
{
    struct orinoco_private *priv = netdev_priv(dev);

    return &priv->stats;
}

-struct iw_statistics *
-orinoco_get_wireless_stats(struct net_device *dev)
+struct iw_statistics *orinoco_get_wireless_stats(struct net_device *dev)
{
    struct orinoco_private *priv = netdev_priv(dev);
    hermes_t *hw = &priv->hw;
@@ -707,8 +708,7 @@
    return wstats;
}

-static void
-orinoco_set_multicast_list(struct net_device *dev)
+static void orinoco_set_multicast_list(struct net_device *dev)
{
    struct orinoco_private *priv = netdev_priv(dev);
    unsigned long flags;
@@ -723,8 +723,7 @@
    orinoco_unlock(priv, &flags);
}

-static int
-orinoco_change_mtu(struct net_device *dev, int new_mtu)
+static int orinoco_change_mtu(struct net_device *dev, int new_mtu)
{
    struct orinoco_private *priv = netdev_priv(dev);

@@ -744,8 +743,7 @@
/* Tx path */
/*****

```

```

-static int
-orinoco_xmit(struct sk_buff *skb, struct net_device *dev)
+static int orinoco_xmit(struct sk_buff *skb, struct net_device *dev)
{
    struct orinoco_private *priv = netdev_priv(dev);
    struct net_device_stats *stats = &priv->stats;
@@ -778,7 +776,6 @@
    printk(KERN_ERR "%s: orinoco_xmit() called while hw_unavailable\n",
           dev->name);
    TRACE_EXIT(dev->name);
-/* BUG(); */
    return 1;
}

@@ -846,7 +843,8 @@
}

/* Round up for odd length packets */
- err = hermes_bap_pwrite(hw, USER_BAP, p, ALIGN(data_len, 2), txfid, data_off);
+ err = hermes_bap_pwrite(hw, USER_BAP, p, ALIGN(data_len, 2),
+ txfid, data_off);
    if (err) {
        printk(KERN_ERR "%s: Error %d writing packet to BAP\n",
               dev->name, err);
@@ -857,10 +855,12 @@
    /* Finally, we actually initiate the send */
    netif_stop_queue(dev);

- err = hermes_docmd_wait(hw, HERMES_CMD_TX | HERMES_CMD_RECL, txfid, NULL);
+ err = hermes_docmd_wait(hw, HERMES_CMD_TX | HERMES_CMD_RECL,
+ txfid, NULL);
    if (err) {
        netif_start_queue(dev);
- printk(KERN_ERR "%s: Error %d transmitting packet\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d transmitting packet\n",
+ dev->name, err);
        stats->tx_errors++;
        goto fail;
    }
@@ -885,7 +885,6 @@
static void __orinoco_ev_alloc(struct net_device *dev, hermes_t *hw)
{
    struct orinoco_private *priv = netdev_priv(dev);
-
    u16 fid = hermes_read_regn(hw, ALLOCFID);

    if (fid != priv->txfid) {
@@ -936,8 +935,7 @@
    hermes_write_regn(hw, TXCOMPLFID, DUMMY_FID);
}

```

```

-static void
-orinoco_tx_timeout(struct net_device *dev)
+static void orinoco_tx_timeout(struct net_device *dev)
{
    struct orinoco_private *priv = netdev_priv(dev);
    struct net_device_stats *stats = &priv->stats;
@@ -959,8 +957,7 @@

/* Does the frame have a SNAP header indicating it should be
 * de-encapsulated to Ethernet-II? */
-static inline int
-is_ethersnap(struct header_struct *hdr)
+static inline int is_ethersnap(struct header_struct *hdr)
{
    /* We de-encapsulate all packets which, a) have SNAP headers
     * (i.e. SSAP=DSAP=0xaa and CTRL=0x3 in the 802.2 LLC header
@@ -972,7 +969,7 @@
}

static inline void orinoco_spy_gather(struct net_device *dev, u_char *mac,
- int level, int noise)
+ int level, int noise)
{
    struct orinoco_private *priv = netdev_priv(dev);
    int i;
@@ -988,10 +985,9 @@
}

-void
-orinoco_stat_gather(struct net_device *dev,
- struct sk_buff *skb,
- struct hermes_rx_descriptor *desc)
+void orinoco_stat_gather(struct net_device *dev,
+ struct sk_buff *skb,
+ struct hermes_rx_descriptor *desc)
{
    struct orinoco_private *priv = netdev_priv(dev);

@@ -1037,7 +1033,7 @@
}

    status = le16_to_cpu(desc.status);
-
+
    if (status & HERMES_RXSTAT_ERR) {
        if (status & HERMES_RXSTAT_UNDECRYPTABLE) {
            wstats->discard.code++;
@@ -1100,9 +1096,9 @@
    * For some reason, the SNAP frames sent by LinkSys APs

```

Linux–Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

```
* are not properly recognised by most firmwares.
* So, check ourselves */
– if(((status & HERMES_RXSTAT_MSGTYPE) == HERMES_RXSTAT_1042) ||
+ ((status & HERMES_RXSTAT_MSGTYPE) == HERMES_RXSTAT_TUNNEL) ||
– is_ethersnap(&hdr)) {
+ if (((status & HERMES_RXSTAT_MSGTYPE) == HERMES_RXSTAT_1042) ||
+ ((status & HERMES_RXSTAT_MSGTYPE) == HERMES_RXSTAT_TUNNEL) ||
+ is_ethersnap(&hdr)) {
    /* These indicate a SNAP within 802.2 LLC within
    802.11 frame which we'll need to de–encapsulate to
    the original EthernetII frame. */
@@ –1268,7 +1264,7 @@
    case HERMES_INQ_LINKSTATUS: {
        struct hermes_linkstatus linkstatus;
        u16 newstatus;
–
+
        if (len != sizeof(linkstatus)) {
            printk(KERN_WARNING "%s: Unexpected size for linkstatus frame (%d bytes)\n",
                dev->name, len);
@@ –1296,8 +1292,8 @@
        }
        break;
        default:
– printk(KERN_DEBUG "%s: Unknown information frame received (type %04x).\n",
– dev->name, type);
+ printk(KERN_DEBUG "%s: Unknown information frame received "
+ "(type %04x).\n", dev->name, type);
        /* We don't actually do anything about it */
        break;
    }
@@ –1487,7 +1483,8 @@
    }

    /* Write the index of the key used in transmission */
– err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFWEPDEFAULTKEYID,
+ err = hermes_write_wordrec(hw, USER_BAP,
+ HERMES_RID_CNFWEPDEFAULTKEYID,
+ priv->tx_key);
        if (err)
            return err;
@@ –1507,11 +1504,12 @@

        err = hermes_write_wordrec(hw, USER_BAP,
– HERMES_RID_CNFAUTHENTICATION, auth_flag);
+ HERMES_RID_CNFAUTHENTICATION,
+ auth_flag);
        if (err)
            return err;
    }
}
```

```

-
+
    /* Master WEP setting : on/off */
    err = hermes_write_wordrec(hw, USER_BAP,
        HERMES_RID_CNFWEPFLAGS_INTERSIL,
@@ -1543,14 +1541,17 @@
    err = hermes_write_ltv(hw, USER_BAP, HERMES_RID_CNFWOWNMACADDR,
        HERMES_BYTES_TO_RECLEN(ETH_ALEN), dev->dev_addr);
    if (err) {
- printk(KERN_ERR "%s: Error %d setting MAC address\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d setting MAC address\n",
+ dev->name, err);
        return err;
    }

    /* Set up the link mode */
- err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFPORRTYPE, priv->port_type);
+ err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFPORRTYPE,
+ priv->port_type);
    if (err) {
- printk(KERN_ERR "%s: Error %d setting port type\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d setting port type\n",
+ dev->name, err);
        return err;
    }

    /* Set the channel/frequency */
@@ -1559,14 +1560,17 @@
    if (priv->createibss)
        priv->channel = 10;
    }
- err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFWOWNCHANNEL, priv->channel);
+ err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFWOWNCHANNEL,
+ priv->channel);
    if (err) {
- printk(KERN_ERR "%s: Error %d setting channel\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d setting channel\n",
+ dev->name, err);
        return err;
    }

    if (priv->has_ibss) {
- err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFCREATEIBSS,
+ err = hermes_write_wordrec(hw, USER_BAP,
+ HERMES_RID_CNFCREATEIBSS,
        priv->createibss);
    if (err) {
        printk(KERN_ERR "%s: Error %d setting CREATEIBSS\n", dev->name, err);
@@ -1575,8 +1579,8 @@

    if ((strlen(priv->desired_essid) == 0) && (priv->createibss)
        && (!priv->has_ibss_any)) {

```

Linux-Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

```

- printk(KERN_WARNING "%s: This firmware requires an \
-ESSID in IBSS-Ad-Hoc mode.\n", dev->name);
+ printk(KERN_WARNING "%s: This firmware requires an "
+ "ESSID in IBSS-Ad-Hoc mode.\n", dev->name);
        /* With wlan_cs, in this case, we would crash.
        * hopefully, this driver will behave better...
        * Jean II */
@@ -1591,14 +1595,16 @@
        HERMES_BYTES_TO_RECLEN(strlen(priv->desired_essid)+2),
        &idbuf);

    if (err) {
- printk(KERN_ERR "%s: Error %d setting OWNSSID\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d setting OWNSSID\n",
+ dev->name, err);
        return err;
    }
    err = hermes_write_ltv(hw, USER_BAP, HERMES_RID_CNFDESIREDSSID,
        HERMES_BYTES_TO_RECLEN(strlen(priv->desired_essid)+2),
        &idbuf);

    if (err) {
- printk(KERN_ERR "%s: Error %d setting DESIREDSSID\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d setting DESIREDSSID\n",
+ dev->name, err);
        return err;
    }

@@ -1609,26 +1615,31 @@
        HERMES_BYTES_TO_RECLEN(strlen(priv->nick)+2),
        &idbuf);

    if (err) {
- printk(KERN_ERR "%s: Error %d setting nickname\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d setting nickname\n",
+ dev->name, err);
        return err;
    }

    /* Set AP density */
    if (priv->has_sensitivity) {
- err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFSYSTEMSCALE,
+ err = hermes_write_wordrec(hw, USER_BAP,
+ HERMES_RID_CNFSYSTEMSCALE,
        priv->ap_density);

        if (err) {
            printk(KERN_WARNING "%s: Error %d setting SYSTEMSCALE. "
- "Disabling sensitivity control\n", dev->name, err);
+ "Disabling sensitivity control\n",
+ dev->name, err);

            priv->has_sensitivity = 0;
        }
    }
}

```

Linux–Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

```
/* Set RTS threshold */
- err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFRSTHRESHOLD, priv->rts_thresh);
+ err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFRSTHRESHOLD,
+ priv->rts_thresh);
    if (err) {
- printk(KERN_ERR "%s: Error %d setting RTS threshold\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d setting RTS threshold\n",
+ dev->name, err);
        return err;
    }

@@ -1642,20 +1653,23 @@
                                HERMES_RID_CNFFRAGMENTATIONTHRESHOLD,
                                priv->frag_thresh);
    if (err) {
- printk(KERN_ERR "%s: Error %d setting fragmentation\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d setting fragmentation\n",
+ dev->name, err);
        return err;
    }

/* Set bitrate */
err = __orinoco_hw_set_bitrate(priv);
if (err) {
- printk(KERN_ERR "%s: Error %d setting bitrate\n", dev->name, err);
+ printk(KERN_ERR "%s: Error %d setting bitrate\n",
+ dev->name, err);
    return err;
}

/* Set power management */
if (priv->has_pm) {
- err = hermes_write_wordrec(hw, USER_BAP, HERMES_RID_CNFPMENABLED,
+ err = hermes_write_wordrec(hw, USER_BAP,
+ HERMES_RID_CNFPMENABLED,
                                priv->pm_on);
    if (err) {
        printk(KERN_ERR "%s: Error %d setting up PM\n",
@@ -1851,7 +1865,7 @@
    if (err)
        /* When the hardware becomes available again, whatever
         * detects that is responsible for re-initializing
- * it. So no need for anything further*/
+ * it. So no need for anything further */
        return;

    netif_stop_queue(dev);
@@ -1870,8 +1884,8 @@
    if (priv->hard_reset)
        err = (*priv->hard_reset)(priv);
```

Linux–Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

```

    if (err) {
- printk(KERN_ERR "%s: orinoco_reset: Error %d performing hard reset\n",
- dev->name, err);
+ printk(KERN_ERR "%s: orinoco_reset: Error %d "
+ "performing hard reset\n", dev->name, err);
        /* FIXME: shutdown of some sort */
        return;
    }
@@ -1930,7 +1944,8 @@
    /* These are used to detect a runaway interrupt situation */
    /* If we get more than MAX_IRQLOOPS_PER_JIFFY iterations in a jiffy,
     * we panic and shut down the hardware */
- static int last_irq_jiffy = 0; /* jiffies value the last time we were called */
+ static int last_irq_jiffy = 0; /* jiffies value the last time
+ * we were called */
    static int loops_this_jiffy = 0;
    unsigned long flags;

@@ -2032,11 +2047,11 @@
        dev->name, err);
        memset(&sta_id, 0, sizeof(sta_id));
    }
+
    le16_to_cpus(&sta_id.id);
    le16_to_cpus(&sta_id.variant);
    le16_to_cpus(&sta_id.major);
    le16_to_cpus(&sta_id.minor);
-
    printk(KERN_DEBUG "%s: Station identity %04x:%04x:%04x:%04x\n",
        dev->name, sta_id.id, sta_id.variant,
        sta_id.major, sta_id.minor);
@@ -2149,8 +2164,7 @@
    }
}

-static int
-orinoco_init(struct net_device *dev)
+static int orinoco_init(struct net_device *dev)
{
    struct orinoco_private *priv = netdev_priv(dev);
    hermes_t *hw = &priv->hw;
@@ -2239,7 +2253,8 @@
    err = hermes_read_wordrec(hw, USER_BAP, HERMES_RID_CNFRTSTHRESHOLD,
        &priv->rts_thresh);

    if (err) {
- printk(KERN_ERR "%s: failed to read RTS threshold!\n", dev->name);
+ printk(KERN_ERR "%s: failed to read RTS threshold!\n",
+ dev->name);
        goto out;
    }
}

```

Linux-Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

@@ -2252,7 +2267,8 @@

```

    err = hermes_read_wordrec(hw, USER_BAP,
HERMES_RID_CNFFRAGMENTATIONTHRESHOLD,
        &priv->frag_thresh);

    if (err) {
- printk(KERN_ERR "%s: failed to read fragmentation settings!\n", dev->name);
+ printk(KERN_ERR "%s: failed to read fragmentation settings!\n",
+ dev->name);
        goto out;
    }

```

@@ -2280,7 +2296,8 @@

```

    /* Preamble setup */
    if (priv->has_preamble) {
- err = hermes_read_wordrec(hw, USER_BAP, HERMES_RID_CNFPREAMBLE_SYMBOL,
+ err = hermes_read_wordrec(hw, USER_BAP,
+ HERMES_RID_CNFPREAMBLE_SYMBOL,
        &priv->preamble);

        if (err)
            goto out;
@@ -2329,18 +2346,20 @@
        return err;
    }

```

```

-struct net_device *alloc_orinocodev(int sizeof_card, int (*hard_reset)(struct orinoco_private *))
+struct net_device *alloc_orinocodev(int sizeof_card,
+ int (*hard_reset)(struct orinoco_private *))
{
    struct net_device *dev;
    struct orinoco_private *priv;

    dev = alloc_etherdev(sizeof(struct orinoco_private) + sizeof_card);
- if (!dev)
+ if (! dev)
    return NULL;
    priv = netdev_priv(dev);
    priv->ndev = dev;
    if (sizeof_card)
- priv->card = (void *)((unsigned long)netdev_priv(dev) + sizeof(struct orinoco_private));
+ priv->card = (void *)((unsigned long)netdev_priv(dev)
+ + sizeof(struct orinoco_private));
    else
        priv->card = NULL;

```

@@ -2700,7 +2719,8 @@

```

    if ((index < 0) || (index >= ORINOCO_MAX_KEYS))
        index = priv->tx_key;
-
+

```

Linux-Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

```
+ /* Adjust key length to a supported value */
    if (erq->length > SMALL_KEY_SIZE) {
        xlen = LARGE_KEY_SIZE;
    } else if (erq->length > 0) {
@@ -2742,14 +2762,14 @@

        if (erq->pointer) {
            priv->keys[index].len = cpu_to_le16(xlen);
- memset(priv->keys[index].data, 0, sizeof(priv->keys[index].data));
+ memset(priv->keys[index].data, 0,
+ sizeof(priv->keys[index].data));
            memcpy(priv->keys[index].data, keybuf, erq->length);
        }
        priv->tx_key = setindex;
        priv->wep_on = enable;
        priv->wep_restrict = restricted;

-
out:
    orinoco_unlock(priv, &flags);

@@ -2975,7 +2995,8 @@
    err = orinoco_lock(priv, &flags);
    if (err)
        return err;
- err = hermes_read_wordrec(hw, USER_BAP, HERMES_RID_CNFSYSTEMSCALE, &val);
+ err = hermes_read_wordrec(hw, USER_BAP,
+ HERMES_RID_CNFSYSTEMSCALE, &val);
    orinoco_unlock(priv, &flags);

    if (err)
@@ -3461,7 +3482,6 @@

    *val = priv->prefer_port3;
    orinoco_unlock(priv, &flags);

-
    return 0;
}

Index: working-2.6/drivers/net/wireless/orinoco_pci.c
=====
--- working-2.6.orig/drivers/net/wireless/orinoco_pci.c 2004-07-28 14:57:52.312664192 +1000
+++ working-2.6/drivers/net/wireless/orinoco_pci.c 2004-07-28 14:57:53.048552320 +1000
@@ -206,18 +206,17 @@
    if (! pci_iorange)
        goto fail;

- /* Usual setup of structures */
+ /* Allocate network device */
    dev = alloc_orinocodev(0, NULL);
    if (! dev) {
```

```

        err = -ENOMEM;
        goto fail;
    }
- priv = netdev_priv(dev);

+ priv = netdev_priv(dev);
    dev->base_addr = (unsigned long) pci_ioaddr;
    dev->mem_start = pci_iorange;
    dev->mem_end = pci_iorange + pci_iolen - 1;
-
    SET_MODULE_OWNER(dev);
    SET_NETDEV_DEV(dev, &pdev->dev);

@@ -238,6 +237,7 @@
        goto fail;
    }
    dev->irq = pdev->irq;
+
    /* Perform a COR reset to start the card */
    if(ornoco_pci_cor_reset(priv) != 0) {
        printk(KERN_ERR "%s: Failed to start the card\n", dev->name);
@@ -255,7 +255,8 @@
        goto fail;
    }

- return 0; /* succeeded */
+ return 0;
+
fail:
    if (dev) {
        if (dev->irq)
@@ -279,7 +280,7 @@

        unregister_netdev(dev);

- if (dev->irq)
+ if (dev->irq)
        free_irq(dev->irq, dev);

    if (priv->hw.iobase)
@@ -357,7 +358,9 @@
    }

    static struct pci_device_id orinoco_pci_pci_id_table[] = {
+ /* Intersil Prism 3 */
        {0x1260, 0x3872, PCI_ANY_ID, PCI_ANY_ID,},
+ /* Intersil Prism 2.5 */
        {0x1260, 0x3873, PCI_ANY_ID, PCI_ANY_ID,},
        {0,},
    };
Index: working-2.6/drivers/net/wireless/orinoco.h

```

```

=====
--- working-2.6.orig/drivers/net/wireless/orinoco.h 2004-07-28 14:56:33.743608504 +1000
+++ working-2.6/drivers/net/wireless/orinoco.h 2004-07-28 14:57:53.049552168 +1000
@@ -12,6 +12,7 @@
#include <linux/netdevice.h>
#include <linux/wireless.h>
#include <linux/version.h>
+
#include "hermes.h"

/* To enable debug messages */
@@ -50,7 +51,6 @@
    hermes_t hw;
    u16 txfid;

-
    /* Capabilities of the hardware/firmware */
    int firmware_type;
#define FIRMWARE_TYPE_AGERE 1
@@ -100,6 +100,10 @@
#define TRACE_ENTER(devname) DEBUG(2, "%s: -> %s()\n", devname, __FUNCTION__);
#define TRACE_EXIT(devname) DEBUG(2, "%s: <- %s()\n", devname, __FUNCTION__);

+/**
+ * Exported prototypes */
+/**
+
+extern struct net_device *alloc_orinocodev(int sizeof_card,
+                                           int (*hard_reset)(struct orinoco_private *));
extern int __orinoco_up(struct net_device *dev);
Index: working-2.6/drivers/net/wireless/orinoco_tmd.c
=====
--- working-2.6.orig/drivers/net/wireless/orinoco_tmd.c 2004-07-28 14:57:52.314663888 +1000
+++ working-2.6/drivers/net/wireless/orinoco_tmd.c 2004-07-28 14:57:53.051551864 +1000
@@ -78,8 +78,7 @@

static char dev_info[] = "orinoco_tmd";

-#define COR_VALUE (COR_LEVEL_REQ | COR_FUNC_ENA | COR_FUNC_ENA) /* Enable PC card
with level triggered irqs and irq requests */
-
+#define COR_VALUE (COR_LEVEL_REQ | COR_FUNC_ENA | COR_FUNC_ENA) /* Enable PC card
with interrupt in level trigger */

static int orinoco_tmd_init_one(struct pci_dev *pdev,
                               const struct pci_device_id *ent)
@@ -115,6 +114,7 @@
    goto fail;
}

+ /* Allocate network device */

```

Linux-Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

```
dev = alloc_orinocodev(0, NULL);
if (! dev) {
    err = -ENOMEM;
@@ -126,16 +126,16 @@
    SET_MODULE_OWNER(dev);
    SET_NETDEV_DEV(dev, &pdev->dev);

- printk(KERN_DEBUG
- "Detected Orinoco/Prism2 TMD device at %s irq:%d, io addr:0x%lx\n",
- pci_name(pdev), pdev->irq, pccard_ioaddr);
+ printk(KERN_DEBUG "Detected Orinoco/Prism2 TMD device "
+ "at %s irq:%d, io addr:0x%lx\n", pci_name(pdev), pdev->irq,
+ pccard_ioaddr);

    hermes_struct_init(&(priv->hw), dev->base_addr,
        HERMES_IO, HERMES_16BIT_REGSPACING);
    pci_set_drvdata(pdev, dev);

- err = request_irq(pdev->irq, orinoco_interrupt, SA_SHIRQ, dev->name,
- dev);
+ err = request_irq(pdev->irq, orinoco_interrupt, SA_SHIRQ,
+ dev->name, dev);
    if (err) {
        printk(KERN_ERR "orinoco_tmd: Error allocating IRQ %d.\n",
            pdev->irq);
@@ -148,9 +148,9 @@
    if (err)
        goto fail;

- return 0; /* succeeded */
+ return 0;

- fail:
+ fail:
    printk(KERN_DEBUG "orinoco_tmd: init_one(), FAIL!\n");

    if (dev) {
Index: working-2.6/drivers/net/wireless/orinoco_plx.c
=====
--- working-2.6.orig/drivers/net/wireless/orinoco_plx.c 2004-07-28 14:57:52.313664040 +1000
+++ working-2.6/drivers/net/wireless/orinoco_plx.c 2004-07-28 14:57:53.054551408 +1000
@@ -1,7 +1,7 @@
/* orinoco_plx.c 0.13e
*
* Driver for Prism II devices which would usually be driven by orinoco_cs,
- * but are connected to the PCI bus by a PLX9052.
+ * but are connected to the PCI bus by a PLX9052.
*
* Copyright (C) 2001 Daniel Barlow <dan AT telent.net>
*
@@ -33,77 +33,80 @@
```

Linux–Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

- * drop me mail with the id and "it works"/"it doesn't work".
- *
- * Note: if everything gets detected fine but it doesn't actually send
 - * or receive packets, your first port of call should probably be to
 - + * or receive packets, your first port of call should probably be to
 - * try newer firmware in the card. Especially if you're doing Ad–Hoc
 - * modes
 - + * modes.
 - *
 - * The actual driving is done by orinoco.c, this is just resource
 - * allocation stuff. The explanation below is courtesy of Ryan Niemi
 - * on the linux–wlan–ng list at
 - * <http://archives.neohapsis.com/archives/dev/linux–wlan/2001–q1/0026.html>
-
- The PLX9052–based cards (WL11000 and several others) are a different
- beast than the usual PCMCIA–based PRISM2 configuration expected by
- wlan–ng. Here's the general details on how the WL11000 PCI adapter
- works:
-
- – Two PCI I/O address spaces, one 0x80 long which contains the PLX9052
- registers, and one that's 0x40 long mapped to the PCMCIA slot I/O
- address space.
-
- – One PCI memory address space, mapped to the PCMCIA memory space
- (containing the CIS).
-
- After identifying the I/O and memory space, you can read through the
- memory space to confirm the CIS's device ID or manufacturer ID to make
- sure it's the expected card. Keep in mind that the PCMCIA spec specifies
- the CIS as the lower 8 bits of each word read from the CIS, so to read the
- bytes of the CIS, read every other byte (0,2,4,...). Passing that test,
- you need to enable the I/O address space on the PCMCIA card via the PCMCIA
- COR register. This is the first byte following the CIS. In my case
- (which may not have any relation to what's on the PRISM2 cards), COR was
- at offset 0x800 within the PCI memory space. Write 0x41 to the COR
- register to enable I/O mode and to select level triggered interrupts. To
- confirm you actually succeeded, read the COR register back and make sure
- it actually got set to 0x41, incase you have an unexpected card inserted.
-
- Following that, you can treat the second PCI I/O address space (the one
- that's not 0x80 in length) as the PCMCIA I/O space.
-
- Note that in the Eumitcom's source for their drivers, they register the
- interrupt as edge triggered when registering it with the Windows kernel. I
- don't recall how to register edge triggered on Linux (if it can be done at
- all). But in some experimentation, I don't see much operational
- difference between using either interrupt mode. Don't mess with the
- interrupt mode in the COR register though, as the PLX9052 wants level
- triggers with the way the serial EEPROM configures it on the WL11000.
-
- There's some other little quirks related to timing that I bumped into, but

–I don't recall right now. Also, there's two variants of the WL11000 I've
–seen, revision A1 and T2. These seem to differ slightly in the timings
–configured in the wait–state generator in the PLX9052. There have also
–been some comments from Eumitcom that cards shouldn't be hot swapped,
–apparently due to risk of cooking the PLX9052. I'm unsure why they
–believe this, as I can't see anything in the design that would really
–cause a problem, except for crashing drivers not written to expect it. And
–having developed drivers for the WL11000, I'd say it's quite tricky to
–write code that will successfully deal with a hot unplug. Very odd things
–happen on the I/O side of things. But anyway, be warned. Despite that,
–I've hot–swapped a number of times during debugging and driver development
–for various reasons (stuck WAIT# line after the radio card's firmware
–locks up).

–
–Hope this is enough info for someone to add PLX9052 support to the wlan–ng
–card. In the case of the WL11000, the PCI ID's are 0x1639/0x0200, with
–matching subsystem ID's. Other PLX9052–based manufacturers other than
–Eumitcom (or on cards other than the WL11000) may have different PCI ID's.

–
–If anyone needs any more specific info, let me know. I haven't had time
–to implement support myself yet, and with the way things are going, might
–not have time for a while..

–
----end of mail----

–*/

+ *

+ * The PLX9052–based cards (WL11000 and several others) are a
+ * different beast than the usual PCMCIA–based PRISM2 configuration
+ * expected by wlan–ng. Here's the general details on how the WL11000
+ * PCI adapter works:

+ *

+ * – Two PCI I/O address spaces, one 0x80 long which contains the
+ * PLX9052 registers, and one that's 0x40 long mapped to the PCMCIA
+ * slot I/O address space.

+ *

+ * – One PCI memory address space, mapped to the PCMCIA memory space
+ * (containing the CIS).

+ *

+ * After identifying the I/O and memory space, you can read through
+ * the memory space to confirm the CIS's device ID or manufacturer ID
+ * to make sure it's the expected card. qKeep in mind that the PCMCIA
+ * spec specifies the CIS as the lower 8 bits of each word read from
+ * the CIS, so to read the bytes of the CIS, read every other byte
+ * (0,2,4,...). Passing that test, you need to enable the I/O address
+ * space on the PCMCIA card via the PCMCIA COR register. This is the
+ * first byte following the CIS. In my case (which may not have any
+ * relation to what's on the PRISM2 cards), COR was at offset 0x800
+ * within the PCI memory space. Write 0x41 to the COR register to
+ * enable I/O mode and to select level triggered interrupts. To
+ * confirm you actually succeeded, read the COR register back and make
+ * sure it actually got set to 0x41, incase you have an unexpected

```
+ * card inserted.
+ *
+ * Following that, you can treat the second PCI I/O address space (the
+ * one that's not 0x80 in length) as the PCMCIA I/O space.
+ *
+ * Note that in the Eumitcom's source for their drivers, they register
+ * the interrupt as edge triggered when registering it with the
+ * Windows kernel. I don't recall how to register edge triggered on
+ * Linux (if it can be done at all). But in some experimentation, I
+ * don't see much operational difference between using either
+ * interrupt mode. Don't mess with the interrupt mode in the COR
+ * register though, as the PLX9052 wants level triggers with the way
+ * the serial EEPROM configures it on the WL11000.
+ *
+ * There's some other little quirks related to timing that I bumped
+ * into, but I don't recall right now. Also, there's two variants of
+ * the WL11000 I've seen, revision A1 and T2. These seem to differ
+ * slightly in the timings configured in the wait–state generator in
+ * the PLX9052. There have also been some comments from Eumitcom that
+ * cards shouldn't be hot swapped, apparently due to risk of cooking
+ * the PLX9052. I'm unsure why they believe this, as I can't see
+ * anything in the design that would really cause a problem, except
+ * for crashing drivers not written to expect it. And having developed
+ * drivers for the WL11000, I'd say it's quite tricky to write code
+ * that will successfully deal with a hot unplug. Very odd things
+ * happen on the I/O side of things. But anyway, be warned. Despite
+ * that, I've hot–swapped a number of times during debugging and
+ * driver development for various reasons (stuck WAIT# line after the
+ * radio card's firmware locks up).
+ *
+ * Hope this is enough info for someone to add PLX9052 support to the
+ * wlan–ng card. In the case of the WL11000, the PCI ID's are
+ * 0x1639/0x0200, with matching subsystem ID's. Other PLX9052–based
+ * manufacturers other than Eumitcom (or on cards other than the
+ * WL11000) may have different PCI ID's.
+ *
+ * If anyone needs any more specific info, let me know. I haven't had
+ * time to implement support myself yet, and with the way things are
+ * going, might not have time for a while..
+ */

#include <linux/config.h>

@@ -134,11 +137,11 @@

static char dev_info[] = "orinoco_plx";

–#define COR_OFFSET (0x3e0 / 2) /* COR attribute offset of Prism2 PC card */
–#define COR_VALUE (COR_LEVEL_REQ | COR_FUNC_ENA) /* Enable PC card with interrupt in level
trigger */
+#define COR_OFFSET (0x3e0/2) /* COR attribute offset of Prism2 PC card */
```

Linux–Kernel: [7/15] orinoco merge preliminaries – comment/whitespace/spelling updates

```
+ #define COR_VALUE (COR_LEVEL_REQ | COR_FUNC_ENA) /* Enable PC card with interrupt in level
trigger */

- #define PLX_INTCSR 0x4c /* Interrupt Control and Status Register */
- #define PLX_INTCSR_INTEN (1<<6) /* Interrupt Enable bit */
+ #define PLX_INTCSR 0x4c /* Interrupt Control & Status Register */
+ #define PLX_INTCSR_INTEN (1<<6) /* Interrupt Enable bit */

static const u16 cis_magic[] = {
    0x0001, 0x0003, 0x0000, 0x0000, 0x00ff, 0x0017, 0x0004, 0x0067
@@ -223,6 +226,7 @@
    goto fail;
}

+ /* Allocate network device */
    dev = alloc_orinocodev(0, NULL);
    if (! dev) {
        err = -ENOMEM;
@@ -234,15 +238,16 @@
    SET_MODULE_OWNER(dev);
    SET_NETDEV_DEV(dev, &pdev->dev);

- printk(KERN_DEBUG
- "Detected Orinoco/Prism2 PLX device at %s irq:%d, io addr:0x%lx\n",
- pci_name(pdev), pdev->irq, pccard_ioaddr);
+ printk(KERN_DEBUG "Detected Orinoco/Prism2 PLX device "
+ "at %s irq:%d, io addr:0x%lx\n", pci_name(pdev), pdev->irq,
+ pccard_ioaddr);

- hermes_struct_init(&(priv->hw), dev->base_addr,
- HERMES_IO, HERMES_16BIT_REGSPACING);
+ hermes_struct_init(&(priv->hw), dev->base_addr, HERMES_IO,
+ HERMES_16BIT_REGSPACING);
    pci_set_drvdata(pdev, dev);

- err = request_irq(pdev->irq, orinoco_interrupt, SA_SHIRQ, dev->name, dev);
+ err = request_irq(pdev->irq, orinoco_interrupt, SA_SHIRQ,
+ dev->name, dev);
    if (err) {
```