

[PATCH] Dell laptop backlight driver

Source: <http://linux.derkeiler.com/Mailing-Lists/Kernel/2007-10/msg10648.html>

- *From:* jack@xxxxxxxxxxxxx
 - *Date:* Sun, 28 Oct 2007 17:12:37 +0100
-

Hello,
this driver implements backlight control on Dell laptops
which use SMI for changing brightness levels.

The driver is INCOMPLETE since it is unable to probe some required parameters
in order to perform backlight control. Such parameters are found in a Dell
proprietary DMI table which should be parsed. For now external tools may be
used to find these parameters by hand. So if you intend to try this out,
FIRST write your laptop model parameters correctly inside the source code
as explained in Documentation/dell-laptop.txt.

Parts of this driver may also be used to provide additional functionalities
similarly to the drivers/misc/*-laptop.c drivers.

regards,
jacopo antonello

```
***** diff follows
--- linux-2.6.23.1/Documentation/dell-laptop.txt 1970-01-01 01:00:00.000000000 +0100
+++ b/Documentation/dell-laptop.txt 2007-10-28 13:45:24.000000000 +0100
@@ -0,0 +1,105 @@
+This driver is EXPERIMENTAL, use it at YOUR OWN RISK.
+
+BEFORE TRYING THIS DRIVER OUT:
+You MUST FILL IN your laptop parameters in the source code. This
+driver is not capable of probing these parameters on its own.
+
+The parameters involved are:
+- IO Port address
+- IO command code
+- Location
+
+These parameters depend (hopefully) on the model of your laptop. Such information
+is found in a private Dell DMI table. You may use dmidecode or libsmbios' dumpCmos
+(http://linux.dell.com/libsmbios/download/libsmbios/). The latter is best since
+it DECODES COMPLETELY Dell's tokens.
+
+***** WITH LIBSMBIOS
+root@here:~# dumpCmos | grep 0x007d
```

[PATCH] Dell laptop backlight driver

+and you will get a line like the following...

+DMI type 0xda Handle 0xda00 CmdIO Port 0x00b2 CmdIO Code 0x0d Type 0x007d Location 0x0000
value 0000

+

+open dell-laptop.c and write the parameters accordingly under the probe_smi_params()

+function

+

+static int probe_smi_params(void)

+{

+ /*

+ * these should be probed by parsing Dell's CMOS table (dmi type 0xda)

+ */

+ smi_params.cmdio_port = 0x00b2; /* libsmbios' dumpCmos CmdIO Port */

+ smi_params.cmdio_code = 0x0d; /* libsmbios' dumpCmos CmdIO Code */

+ smi_params.backlight_location = 0; /* set to dumpCmos Location */

+ smi_params.backlight_value = 0; /* set to dumpCmos Value (unused for backlight) */

+

+ return 0;

+}

+*****

+

+***** WITH DMIDECODE

+(this is HAND parsing!!)

+dmidecode & look for table type 218

+

+Handle 0xDA00, DMI type 218, 125 bytes

+OEM-specific Type

+ Header and Data:

+ DA 7D 00 DA B2 00 0D 5F 0F 37 40 7D 00 00 00 00

+ 00 7E 00 02 00 00 00 40 00 04 00 01 00 41 00 04

+ 00 00 00 90 00 05 00 00 00 91 00 05 00 01 00 92

+ 00 05 00 02 00 45 01 45 01 01 00 44 01 44 01 00

+ 00 00 80 00 80 01 00 00 A0 00 A0 01 00 05 80 05

+ 80 01 00 76 01 76 01 01 00 75 01 75 01 01 00 01

+ F0 01 F0 00 00 02 F0 02 F0 00 00 03 F0 03 F0 00

+ 00 04 F0 04 F0 00 00 FF FF 00 00 00 00

+

+now you have:

+1 byte DA (table type)

+1 byte 7D (table size)

+2 bytes 00 DA (handle)

+2 bytes B2 00 (cmdio port address (swapped))

+1 byte 0D (cmdio code)

+4 bytes 5F 0F 37 40 (supported commands ?)

+

+after that you have triplets terminated by FF FF 00 00 00 00

+in the form (type, location, value)

+

+type | loc | val

+7D 00|00 00|00 00

+7E 00|02 00|00 00

[PATCH] Dell laptop backlight driver

```
+40 00|04 00|01 00
+41 00|04 00|00 00
+90 00|05 00|00 00
+91 00|05 00|01 00
+92 00|05 00|02 00
+45 01|45 01|01 00
+44 01|44 01|00 00
+00 80|00 80|01 00
+00 A0|00 A0|01 00
+05 80|05 80|01 00
+76 01|76 01|01 00
+75 01|75 01|01 00
+01 F0|01 F0|00 00
+02 F0|02 F0|00 00
+03 F0|03 F0|00 00
+04 F0|04 F0|00 00
+end
+FF FF|00 00|00 00
+
+you can read the Location field in type 0x007d (swapped 7D 00)
+
+then modify dell-laptop.c accordingly (remember to swap the lsb with the msb
+n the port address also!)
+
+static int probe_smi_params(void)
+{
+ /*
+ * these should be probed by parsing Dell's CMOS table (dmi type 0xda)
+ */
+ smi_params.cmdio_port = 0x00b2; /* libsmbios' dumpCmos CmdIO Port */
+ smi_params.cmdio_code = 0x0d; /* libsmbios' dumpCmos CmdIO Code */
+ smi_params.backlight_location = 0; /* set to dumpCmos Location */
+ smi_params.backlight_value = 0; /* set to dumpCmos Value (unused for backlight) */
+
+ return 0;
+}
+*****
--- linux-2.6.23.1/drivers/misc/Makefile 2007-10-12 18:43:44.000000000 +0200
+++ b/drivers/misc/Makefile 2007-10-28 13:45:24.000000000 +0100
@@ -15,3 +15,4 @@ obj-$(CONFIG_SGI_IOC4) += ioc4.o
obj-$(CONFIG_SONY_LAPTOP) += sony-laptop.o
obj-$(CONFIG_THINKPAD_ACPI) += thinkpad_acpi.o
obj-$(CONFIG_EEPROM_93CX6) += eeprom_93cx6.o
+obj-$(CONFIG_DELL_LAPTOP) += dell-laptop.o
--- linux-2.6.23.1/drivers/misc/dell-laptop.h 1970-01-01 01:00:00.000000000 +0100
+++ b/drivers/misc/dell-laptop.h 2007-10-28 13:45:24.000000000 +0100
@@ -0,0 +1,116 @@
+/*
+ * dell-laptop.h - Dell laptop extras
+ *
+ *
```

[PATCH] Dell laptop backlight driver

```
+ * Copyright (C) 2007 Jacopo Antonello <jack@xxxxxxxxxxxxxx>
+ *
+ * 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., 51 Franklin Street, Fifth Floor, Boston, MA
+ * 02110-1301, USA.
+ */
+
+#ifndef _DELL_LAPTOP_H_
+#define _DELL_LAPTOP_H_
+
+#include <linux/device.h>
+#include <linux/sysfs.h>
+#include <linux/types.h>
+
+#define SMI_CMD_MAGIC 0x534D4931 /* "SMI1" */
+#define SMI_CMD_ECX 0x42534931 /* "BSI1" */
+
+#define SMI_CMD_SIZE sizeof(struct smi_cmd)
+#define SMI_BUF_SIZE sizeof(struct smi_cmd_buffer)
+#define SMI_IF_SIZE (SMI_CMD_SIZE + SMI_BUF_SIZE)
+
+#define SMI_CLASS_READ_SETTING 0
+#define SMI_CLASS_WRITE_SETTING 1
+
+#define SMI_SELECT_BATTERY 1
+#define SMI_SELECT_AC 2
+
+#define BRIGHTNESS_BATTERY (SMI_SELECT_BATTERY - 1)
+#define BRIGHTNESS_AC (SMI_SELECT_AC - 1)
+
+#define DELL_TABLE_DA 0xda
+#define DELL_CALLIF_ID_BACKLIGHT 0x007d
+
+#define DRIVER_NAME "dell-laptop"
+#define DRIVER_DESCRIPTION "Dell laptop extras"
+#define DRV_PFX "dell-laptop: "
+
+struct smi_cmd {
+    __u32 magic;
+    __u32 ebx;
```

[PATCH] Dell laptop backlight driver

```
+ __u32 ecx;
+ __u16 command_address;
+ __u8 command_code;
+ __u8 reserved;
+ /* __u8 command_buffer[1]; */
+} __attribute__((packed));
+
+struct smi_cmd_buffer
+{
+ __u16 smi_class;
+ __u16 smi_select;
+
+ __u32 cbARG1;
+ __u32 cbARG2;
+ __u32 cbARG3;
+ __u32 cbARG4;
+
+ __s32 cbRES1;
+ __s32 cbRES2;
+ __s32 cbRES3;
+ __s32 cbRES4;
+} __attribute__((packed));
+
+struct smi_params_t {
+ struct mutex lock;
+
+ u16 cmdio_port;
+ u8 cmdio_code;
+ u32 supported_cmds;
+
+ u16 backlight_location;
+ u16 backlight_value;
+};
+
+enum backlight_drive_mode {
+ BACKLIGHT_DRIVE_BOTH,
+ BACKLIGHT_DRIVE_CURRENT,
+ BACKLIGHT_DRIVE_BATTERY,
+ BACKLIGHT_DRIVE_AC
+};
+
+static struct platform_driver dell_driver = {
+ .driver = {
+ .name = DRIVER_NAME,
+ .owner = THIS_MODULE,
+ }
+};
+
+static struct backlight_device *dell_backlight_dev;
+static struct platform_device *dell_platform_dev;
+
```

[PATCH] Dell laptop backlight driver

```
+static u8 *smi_if;
+static dma_addr_t smi_handle;
+
+static struct smi_params_t smi_params;
+
+static enum backlight_drive_mode backlight_drive_mode;
+static u32 brightness_vals[2];
+static u32 brightness_min_value = 0;
+
+#endif
--- linux-2.6.23.1/drivers/misc/dell-laptop.c 1970-01-01 01:00:00.000000000 +0100
+++ b/drivers/misc/dell-laptop.c 2007-10-28 13:45:24.000000000 +0100
@@ -0,0 +1,651 @@
+/*
+ * dell-laptop.c - Dell laptop extras
+ *
+ * This driver implements only backlight control for now.
+ * This driver assumes that you are not using bios password
+ * protection and does not work if it is enabled.
+ *
+ * This driver is EXPERIMENTAL and it is not yet complete. In
+ * particular it lacks proper probing for the hardware
+ * it is meant to control.
+ *
+ * WARNING: you MUST fill in your laptop CMOS parameters, by
+ * modifying the default values found in probe_smi_params().
+ * Failing to do so could cause unpredictable effects on your
+ * laptop. In order to do this you may use the dumpCmos utility
+ * found in libsmbios (http://linux.dell.com/libsmbios/download/libsmbios/)
+ *
+ * Parts of this driver were inspired by dcdbas.c and libsmbios
+ * (http://linux.dell.com/libsmbios/download/libsmbios/)
+ *
+ * Copyright (C) 2007 Jacopo Antonello <jack@xxxxxxxxxxxxxx>
+ *
+ * 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., 51 Franklin Street, Fifth Floor, Boston, MA
+ * 02110-1301, USA.
+ */
+
```

[PATCH] Dell laptop backlight driver

```
+ #define DRIVER_VERSION "0.1"
+
+ /*
+  * Changelog:
+  * 2007-12-22 0.01 initial release
+  * support for backlight on Dell M70 laptops
+  * TODO fix framebuffer support
+  * TODO add cmos token lookup
+  */
+
+ #include <linux/kernel.h>
+ #include <linux/module.h>
+ #include <linux/platform_device.h>
+ #include <linux/dma-mapping.h>
+ #include <linux/mutex.h>
+ #include <linux/sched.h>
+ #include <linux/backlight.h>
+
+ #include "dell-laptop.h"
+
+ MODULE_DESCRIPTION(DRIVER_DESCRIPTION " (version " DRIVER_VERSION ")");
+ MODULE_VERSION(DRIVER_VERSION);
+ MODULE_AUTHOR("jacopo antonello <jack at antonello.org>");
+ MODULE_LICENSE("GPL");
+
+ /* -----
+  * smi
+  * -----
+  */
+
+ static int probe_smi_params(void)
+ {
+ /*
+  * these should be probed by parsing Dell's CMOS table (dmi type 0xda)
+  */
+ smi_params.cmdio_port = 0x00b2; /* libsmbios' dumpCmos CmdIO Port */
+ smi_params.cmdio_code = 0x0d; /* libsmbios' dumpCmos CmdIO Code */
+ smi_params.backlight_location = 0; /* set to dumpCmos Location */
+ smi_params.backlight_value = 0; /* set to dumpCmos Value (unused for backlight) */
+
+ return 0;
+ }
+
+ static int trigger_smi(u16 ioport, u8 iocode, struct smi_cmd_buffer *parms)
+ {
+ struct smi_cmd *cmd = (struct smi_cmd *)smi_if;
+ struct smi_cmd_buffer *buf = (struct smi_cmd_buffer *) (smi_if + SMI_CMD_SIZE);
+ cpumask_t old_mask;
+ int ret = 0;
+
+ mutex_lock(&smi_params.lock);
```

[PATCH] Dell laptop backlight driver

```
+
+ cmd->magic = SMI_CMD_MAGIC;
+ cmd->ebx = (u32)virt_to_phys(buf);
+ cmd->ecx = SMI_CMD_ECX;
+ cmd->command_address = ioport; /* libsmbios' dumpCmos CmdIO Port */
+ cmd->command_code = iocode; /* libsmbios' dumpCmos CmdIO Code */
+ cmd->reserved = 0;
+
+ buf->smi_class = parms->smi_class; /* 0 read, 1 write */
+ buf->smi_select = parms->smi_select; /* 1 battery mode, 2 ac mode */
+
+ buf->cbARG1 = parms->cbARG1; /* set to dumpCmos Location */
+ buf->cbARG2 = parms->cbARG2;
+ buf->cbARG3 = parms->cbARG3;
+ buf->cbARG4 = parms->cbARG4;
+
+ buf->cbRES1 = parms->cbRES1; /* set to -1 to tell errors */
+ buf->cbRES2 = parms->cbRES2;
+ buf->cbRES3 = parms->cbRES3;
+ buf->cbRES4 = parms->cbRES4;
+
+ /* dcdbas.c */
+ old_mask = current->cpus_allowed;
+ set_cpus_allowed(current, cpumask_of_cpu(0));
+ if (smp_processor_id() != 0) {
+ dev_dbg(&dell_backlight_dev->dev, "%s: failed to get CPU 0\n",
+ __FUNCTION__);
+ ret = -EBUSY;
+ goto out;
+ }
+
+ asm volatile (
+ "outb %b0,%w1"
+ : /* no output args */
+ : "a" (cmd->command_code),
+ "d" (cmd->command_address),
+ "b" (cmd->ebx),
+ "c" (cmd->ecx)
+ : "memory"
+ );
+
+ switch( buf->cbRES1 ) {
+ case -6: /* output buffer not large enough */
+ case -5: /* output buffer format error */
+ case -3: /* unhandled smi call */
+ case -2: /* unsupported smi call */
+ case -1: /* bios returned error for smi call */
+ ret = -EBUSY;
+ printk(KERN_WARNING DRV_PFX "SMI failed\n");
+ goto out;
+ }
```

[PATCH] Dell laptop backlight driver

```
+ + /* copy back results */
+ parms->smi_class = buf->smi_class;
+ parms->smi_select = buf->smi_select;
+
+ parms->cbARG1 = buf->cbARG1;
+ parms->cbARG2 = buf->cbARG2;
+ parms->cbARG3 = buf->cbARG3;
+ parms->cbARG4 = buf->cbARG4;
+
+ parms->cbRES1 = buf->cbRES1;
+ parms->cbRES2 = buf->cbRES2;
+ parms->cbRES3 = buf->cbRES3;
+ parms->cbRES4 = buf->cbRES4;
+
+out:
+ set_cpus_allowed(current, old_mask);
+
+ mutex_unlock(&smi_params.lock);
+
+ return ret;
+}
+
+/* -----
+ * brightness
+ * -----
+ */
+
+/* use read_brightness(SMI_SELECT_*) */
+static void read_brightness(int select)
+{
+ struct smi_cmd_buffer buf = {
+ .smi_class = SMI_CLASS_READ_SETTING,
+ .smi_select = select,
+
+ .cbARG1 = smi_params.backlight_location,
+ .cbARG2 = 0,
+ .cbARG3 = 0,
+ .cbARG4 = 0,
+
+ .cbRES1 = -1,
+ .cbRES2 = 0,
+ .cbRES3 = 0,
+ .cbRES4 = 0
+ };
+
+ if (trigger_smi(smi_params.cmdio_port, smi_params.cmdio_code, &buf)) {
+ printk(KERN_ERR DRV_PFX "unable to read brightness levels\n");
+ return;
+ }
+
+ brightness_vals[select - 1] = buf.cbRES2;
```

[PATCH] Dell laptop backlight driver

```
+ printk(KERN_DEBUG DRV_PFX " brightness_vals[ %d ]: %d\n", select - 1,
+ brightness_vals[select - 1]);
+ brightness_min_value = buf.cbRES3;
+ printk(KERN_DEBUG DRV_PFX " brightness_min_value: %d\n", brightness_min_value);
+ dell_backlight_dev->props.max_brightness = buf.cbRES4;
+ printk(KERN_DEBUG DRV_PFX " props.max_brightness: %d\n",
+ dell_backlight_dev->props.max_brightness);
+ }
+
+ /* use write_brightness(SMI_SELECT_*) */
+static void write_brightness(int select)
+{
+ struct smi_cmd_buffer buf = {
+ .smi_class = SMI_CLASS_WRITE_SETTING,
+ .smi_select = select,
+
+ .cbARG1 = smi_params.backlight_location,
+ .cbARG2 = brightness_vals[select - 1],
+ .cbARG3 = 0,
+ .cbARG4 = 0,
+
+ .cbRES1 = -1,
+ .cbRES2 = 0,
+ .cbRES3 = 0,
+ .cbRES4 = 0
+ };
+
+ if (trigger_smi(smi_params.cmdio_port, smi_params.cmdio_code, &buf)) {
+ printk(KERN_ERR DRV_PFX "unable to write brightness\n");
+ return;
+ }
+
+ if (buf.cbRES2 != brightness_vals[select - 1]) {
+ printk(KERN_ERR DRV_PFX "brightness value unchanged\n");
+ }
+
+ brightness_vals[select - 1] = buf.cbRES2;
+ printk(KERN_DEBUG DRV_PFX " brightness_vals[ %d ]: %d\n", select - 1,
+ brightness_vals[select - 1]);
+ }
+
+static int dell_backlight_update_status(struct backlight_device *dev)
+{
+ dev->props.brightness = dev->props.brightness < brightness_min_value ?
+ brightness_min_value : dev->props.brightness > dev->props.max_brightness ?
+ dev->props.max_brightness : dev->props.brightness;
+
+ switch(backlight_drive_mode) {
+ case BACKLIGHT_DRIVE_BOTH:
+ brightness_vals[BRIGHTNESS_BATTERY] = dev->props.brightness;
+ brightness_vals[BRIGHTNESS_AC] = dev->props.brightness;
+ }
```

[PATCH] Dell laptop backlight driver

```
+ write_brightness(SMI_SELECT_BATTERY);
+ write_brightness(SMI_SELECT_AC);
+ break;
+ case BACKLIGHT_DRIVE_AC:
+ brightness_vals[BRIGHTNESS_AC] = dev->props.brightness;
+ write_brightness(SMI_SELECT_AC);
+ break;
+ case BACKLIGHT_DRIVE_BATTERY:
+ brightness_vals[BRIGHTNESS_BATTERY] = dev->props.brightness;
+ write_brightness(SMI_SELECT_BATTERY);
+ break;
+ case BACKLIGHT_DRIVE_CURRENT:
+ printk(KERN_ERR DRV_PFX "BACKLIGHT_DRIVER_CURRENT unimplemented\n");
+ default:
+ printk(KERN_ERR DRV_PFX "unknown backlight_drive_mode\n");
+ }
+
+ return dev->props.brightness;
+}
+
+static void update_from_brightness_vals(void)
+{
+ switch(backlight_drive_mode) {
+ case BACKLIGHT_DRIVE_BOTH:
+ dell_backlight_dev->props.brightness = brightness_vals[BRIGHTNESS_BATTERY];
+ if (brightness_vals[BRIGHTNESS_AC] !=
+ brightness_vals[BRIGHTNESS_BATTERY]) {
+ brightness_vals[BRIGHTNESS_AC] = brightness_vals[BRIGHTNESS_BATTERY];
+ write_brightness(SMI_SELECT_AC); /* force ac to the same value */
+ }
+ break;
+ case BACKLIGHT_DRIVE_AC:
+ dell_backlight_dev->props.brightness = brightness_vals[BRIGHTNESS_AC];
+ break;
+ case BACKLIGHT_DRIVE_BATTERY:
+ dell_backlight_dev->props.brightness = brightness_vals[BRIGHTNESS_BATTERY];
+ break;
+ case BACKLIGHT_DRIVE_CURRENT:
+ printk(KERN_ERR DRV_PFX "BACKLIGHT_DRIVER_CURRENT unimplemented\n");
+ default:
+ printk(KERN_ERR DRV_PFX "unknown backlight_drive_mode\n");
+ }
+ }
+
+static int dell_backlight_get_brightness(struct backlight_device *dev)
+{
+ /* is locking needed? */
+ mutex_lock(&dell_backlight_dev->update_lock);
+
+ read_brightness(SMI_SELECT_BATTERY);
+ read_brightness(SMI_SELECT_AC);
```

[PATCH] Dell laptop backlight driver

```
+
+ update_from_brightness_vals();
+
+ mutex_unlock(&dell_backlight_dev->update_lock);
+
+ return dev->props.brightness;
+}
+
+static struct backlight_ops dell_backlight_ops = {
+ .update_status = dell_backlight_update_status,
+ .get_brightness = dell_backlight_get_brightness
+};
+
+static int dell_backlight_init(struct device *dev)
+{
+ dell_backlight_dev = backlight_device_register(
+ "dell-laptop",
+ NULL,
+ NULL,
+ &dell_backlight_ops);
+
+ if (IS_ERR(dell_backlight_dev)) {
+ dell_backlight_dev = NULL;
+ return PTR_ERR(dell_backlight_dev);
+ }
+
+ backlight_drive_mode = BACKLIGHT_DRIVE_AC;
+
+ dell_backlight_get_brightness(dell_backlight_dev);
+
+ return 0;
+}
+
+/* -----
+ * sysfs
+ * -----
+ */
+
+/* show */
+static ssize_t show_bl_drive_mode(struct device *dev,
+ struct device_attribute *attr, char *buf)
+{
+ char *mode = "unknown";
+
+ switch(backlight_drive_mode) {
+ case BACKLIGHT_DRIVE_BOTH:
+ mode = "BACKLIGHT_DRIVE_BOTH";
+ break;
+ case BACKLIGHT_DRIVE_CURRENT:
+ mode = "BACKLIGHT_DRIVE_CURRENT";
+ break;
```

[PATCH] Dell laptop backlight driver

```
+ case BACKLIGHT_DRIVE_BATTERY:
+ mode = "BACKLIGHT_DRIVE_BATTERY";
+ break;
+ case BACKLIGHT_DRIVE_AC:
+ mode = "BACKLIGHT_DRIVE_AC";
+ break;
+ }
+
+ return snprintf(buf, PAGE_SIZE, "%s\n", mode);
+}
+
+static ssize_t show_bl_drive_mode_list(struct device *dev,
+ struct device_attribute *attr, char *buf)
+{
+ int len = 0;
+
+ len = snprintf(buf, PAGE_SIZE, "%d:BACKLIGHT_DRIVE_BOTH\n", BACKLIGHT_DRIVE_BOTH);
+ len += snprintf(buf + len, PAGE_SIZE - len, "%d:BACKLIGHT_DRIVE_CURRENT\n",
+ BACKLIGHT_DRIVE_CURRENT);
+ len += snprintf(buf + len, PAGE_SIZE - len, "%d:BACKLIGHT_DRIVE_BATTERY\n",
+ BACKLIGHT_DRIVE_BATTERY);
+ len += snprintf(buf + len, PAGE_SIZE - len, "%d:BACKLIGHT_DRIVE_AC\n",
+ BACKLIGHT_DRIVE_AC);
+
+ return len;
+}
+
+static ssize_t show_cmdio_port(struct device *dev,
+ struct device_attribute *attr, char *buf)
+{
+ return snprintf(buf, PAGE_SIZE, "0x%04x\n", smi_params.cmdio_port);
+}
+
+static ssize_t show_cmdio_code(struct device *dev,
+ struct device_attribute *attr, char *buf)
+{
+ return snprintf(buf, PAGE_SIZE, "0x%02x\n", smi_params.cmdio_code);
+}
+
+static ssize_t show_bl_location(struct device *dev,
+ struct device_attribute *attr, char *buf)
+{
+ return snprintf(buf, PAGE_SIZE, "0x%04x\n", smi_params.backlight_location);
+}
+
+static ssize_t show_bl_battery_val(struct device *dev,
+ struct device_attribute *attr, char *buf)
+{
+ return snprintf(buf, PAGE_SIZE, "%d\n",
+ brightness_vals[BRIGHTNESS_BATTERY]);
+}
```

[PATCH] Dell laptop backlight driver

```
+
+static ssize_t show_bl_ac_val(struct device *dev,
+ struct device_attribute *attr, char *buf)
+{
+ return snprintf(buf, PAGE_SIZE, "%d\n",
+ brightness_vals[BRIGHTNESS_AC]);
+}
+
+static ssize_t show_bl_min_val(struct device *dev,
+ struct device_attribute *attr, char *buf)
+{
+ return snprintf(buf, PAGE_SIZE, "%d\n",
+ brightness_min_value);
+}
+
+static ssize_t show_bl_max_val(struct device *dev,
+ struct device_attribute *attr, char *buf)
+{
+ return snprintf(buf, PAGE_SIZE, "%d\n",
+ dell_backlight_dev->props.max_brightness);
+}
+
+/* store */
+static ssize_t store_bl_drive_mode(struct device *dev,
+ struct device_attribute *attr, const char *buf, size_t count)
+{
+ char *endp;
+ int mode = simple_strtoul(buf, &endp, 0);
+
+ switch(mode) {
+ case BACKLIGHT_DRIVE_BOTH:
+ case BACKLIGHT_DRIVE_BATTERY:
+ case BACKLIGHT_DRIVE_AC:
+ break;
+ case BACKLIGHT_DRIVE_CURRENT:
+ default:
+ return -ENXIO;
+ }
+
+ mutex_lock(&dell_backlight_dev->update_lock);
+ backlight_drive_mode = mode;
+ mutex_unlock(&dell_backlight_dev->update_lock);
+
+ return count;
+}
+
+static ssize_t store_cmdio_port(struct device *dev,
+ struct device_attribute *attr, const char *buf, size_t count)
+{
+ char *endp;
+ int port = simple_strtoul(buf, &endp, 0);
```

[PATCH] Dell laptop backlight driver

```
+
+ if (port <= 0 || port >= 0xffff)
+ return -ENXIO;
+
+ mutex_lock(&smi_params.lock);
+ smi_params.cmdio_port = port;
+ mutex_unlock(&smi_params.lock);
+
+ return count;
+}
+
+static ssize_t store_cmdio_code(struct device *dev,
+ struct device_attribute *attr, const char *buf, size_t count)
+{
+ char *endp;
+ int code = simple_strtoul(buf, &endp, 0);
+
+ if (code <= 0 || code >= 0xff)
+ return -ENXIO;
+
+ mutex_lock(&smi_params.lock);
+ smi_params.cmdio_code = code;
+ mutex_unlock(&smi_params.lock);
+
+ return count;
+}
+
+static ssize_t store_bl_location(struct device *dev,
+ struct device_attribute *attr, const char *buf, size_t count)
+{
+ char *endp;
+ int location = simple_strtoul(buf, &endp, 0);
+
+ if (location <= 0 || location >= 0xffff)
+ return -ENXIO;
+
+ mutex_lock(&dell_backlight_dev->update_lock);
+ smi_params.backlight_location = location;
+ mutex_unlock(&dell_backlight_dev->update_lock);
+
+ return count;
+}
+
+static ssize_t store_bl_battery_val(struct device *dev,
+ struct device_attribute *attr, const char *buf, size_t count)
+{
+ char *endp;
+ int level = simple_strtoul(buf, &endp, 0);
+
+ if (level < brightness_min_value ||
+ level > dell_backlight_dev->props.max_brightness)
```

[PATCH] Dell laptop backlight driver

```
+ return -ENXIO;
+
+ mutex_lock(&dell_backlight_dev->update_lock);
+ brightness_vals[BRIGHTNESS_BATTERY] = level;
+ write_brightness(SMI_SELECT_BATTERY);
+ update_from_brightness_vals();
+ mutex_unlock(&dell_backlight_dev->update_lock);
+
+ return count;
+}
+
+static ssize_t store_bl_ac_val(struct device *dev,
+ struct device_attribute *attr, const char *buf, size_t count)
+{
+ char *endp;
+ int level = simple_strtoul(buf, &endp, 0);
+
+ if (level < brightness_min_value ||
+ level > dell_backlight_dev->props.max_brightness)
+ return -ENXIO;
+
+ mutex_lock(&dell_backlight_dev->update_lock);
+ brightness_vals[BRIGHTNESS_AC] = level;
+ write_brightness(SMI_SELECT_AC);
+ if (backlight_drive_mode == BACKLIGHT_DRIVE_BOTH) {
+ /* update_from_brightness_vals forces level to battery mode */
+ brightness_vals[BRIGHTNESS_BATTERY] = level;
+ }
+ update_from_brightness_vals();
+ mutex_unlock(&dell_backlight_dev->update_lock);
+
+ return count;
+}
+
+/* static */
+#define DELL_DEVICE_ATTR(_name, _mode, _show, _store) \
+ struct device_attribute dev_attr_##_name = { \
+ .attr = { \
+ .name = __stringify(_name), \
+ .mode = _mode \
+ }, \
+ .show = _show, \
+ .store = _store \
+ }
+
+static DELL_DEVICE_ATTR(backlight_drive_mode, 0666, show_bl_drive_mode, store_bl_drive_mode);
+static DELL_DEVICE_ATTR(backlight_drive_mode_list, 0444, show_bl_drive_mode_list, NULL);
+static DELL_DEVICE_ATTR(cmdio_port, 0600, show_cmdio_port, store_cmdio_port);
+static DELL_DEVICE_ATTR(cmdio_code, 0600, show_cmdio_code, store_cmdio_code);
+static DELL_DEVICE_ATTR(backlight_location, 0600, show_bl_location, store_bl_location);
+static DELL_DEVICE_ATTR(backlight_battery_val, 0666, show_bl_battery_val, store_bl_battery_val);
```

[PATCH] Dell laptop backlight driver

```
+static DELL_DEVICE_ATTR(backlight_ac_val, 0666, show_bl_ac_val, store_bl_ac_val);
+static DELL_DEVICE_ATTR(backlight_min_val, 0444, show_bl_min_val, NULL);
+static DELL_DEVICE_ATTR(backlight_max_val, 0444, show_bl_max_val, NULL);
+
+static struct attribute *dell_laptop_attrs[] = {
+ &dev_attr_backlight_drive_mode.attr,
+ &dev_attr_backlight_drive_mode_list.attr,
+ &dev_attr_cmdio_port.attr,
+ &dev_attr_cmdio_code.attr,
+ &dev_attr_backlight_location.attr,
+ &dev_attr_backlight_battery_val.attr,
+ &dev_attr_backlight_ac_val.attr,
+ &dev_attr_backlight_min_val.attr,
+ &dev_attr_backlight_max_val.attr,
+ NULL
+};
+
+static struct attribute_group dell_laptop_attr_group = {
+ .attrs = dell_laptop_attrs
+};
+
+/* -----
+ * init / cleanup
+ * -----
+*/
+
+static int __init dell_laptop_init(void)
+{
+ int error;
+
+ error = platform_driver_register(&dell_driver);
+ if (error)
+ return error;
+
+ dell_platform_dev = platform_device_alloc(DRIVER_NAME, -1);
+ if (!dell_platform_dev) {
+ error = -ENOMEM;
+ goto unregister_platform;
+ }
+
+ error = platform_device_add(dell_platform_dev);
+ if (error)
+ goto put_platform;
+
+ /* 32-bit address space */
+ dell_platform_dev->dev.coherent_dma_mask = DMA_32BIT_MASK;
+ dell_platform_dev->dev.dma_mask = &dell_backlight_dev->dev.coherent_dma_mask;
+
+ /* init smi mutex */
+ mutex_init(&smi_params.lock);
+
+
+}
```

[PATCH] Dell laptop backlight driver

```
+ probe_smi_params();
+
+ /* allocate smi buffer */
+ smi_if = dma_alloc_coherent(&dell_platform_dev->dev,
+ SMI_IF_SIZE, &smi_handle, GFP_KERNEL);
+ if (!smi_if) {
+ error = -ENOMEM;
+ goto put_platform;
+ }
+
+ error = dell_backlight_init(&dell_platform_dev->dev);
+ if (error)
+ goto fail_backlight;
+ error = sysfs_create_group(&dell_platform_dev->dev.kobj, &dell_laptop_attr_group);
+ if (error) {
+ error = -ENODEV;
+ goto fail_sysfs;
+ }
+
+ printk(KERN_INFO DRV_PFX "%s (ver: %s) loaded\n", DRIVER_NAME, DRIVER_VERSION);
+
+ return 0;
+
+
+fail_sysfs:
+ /* */
+fail_backlight:
+ /* */
+put_platform:
+ platform_device_put(dell_platform_dev);
+unregister_platform:
+ platform_driver_unregister(&dell_driver);
+
+ return error;
+}
+
+static void __exit dell_laptop_exit(void)
+{
+ backlight_device_unregister(dell_backlight_dev);
+
+ dma_free_coherent(&dell_platform_dev->dev,
+ SMI_IF_SIZE, smi_if, smi_handle);
+ smi_if = 0;
+ smi_handle = 0;
+
+ sysfs_remove_group(&dell_platform_dev->dev.kobj, &dell_laptop_attr_group);
+
+ platform_device_unregister(dell_platform_dev);
+ platform_driver_unregister(&dell_driver);
+
+}
```

[PATCH] Dell laptop backlight driver

```
+ printk(KERN_INFO DRV_PFX "%s (ver: %s) unloaded\n",DRIVER_NAME,DRIVER_VERSION);
+}
+
+module_init(dell_laptop_init);
+module_exit(dell_laptop_exit);
+
--- linux-2.6.23.1/drivers/misc/Kconfig 2007-10-12 18:43:44.000000000 +0200
+++ b/drivers/misc/Kconfig 2007-10-28 15:45:41.000000000 +0100
@@ -130,6 +130,21 @@ config MSI_LAPTOP
```

If you have an MSI S270 laptop, say Y or M here.

```
+config DELL_LAPTOP
+ tristate "Dell Laptop Extras (EXPERIMENTAL)"
+ depends on X86
+ depends on EXPERIMENTAL
+ depends on BACKLIGHT_CLASS_DEVICE
+ ---help---
+ This driver implements LCD backlight control for Dell laptops. BEFORE
+ using this driver, please fill in your laptop model parameters as described
+ in dell-laptop.txt. Using this driver without doing this operation
+ may have UNPREDICTABLE effects on your laptop. You have been warned!
+
+ Just read <file:Documentation/dell-laptop.txt> first.
+
+ If you have a Dell laptop, say Y or M here.
+
config SONY_LAPTOP
tristate "Sony Laptop Extras"
depends on X86 && ACPI
```

-

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/>