thinkpad-acpi.txt
ThinkPad ACPI Extras Driver

Version 0.24
December 11th, 2009

Borislav Deianov <borislav@users.sf.net>
Henrique de Moraes Holschuh <hmh@hmh.eng.br>
http://ibm-acpi.sf.net/

This is a Linux driver for the IBM and Lenovo ThinkPad laptops. It supports various features of these laptops which are accessible through the ACPI and ACPI EC framework, but not otherwise fully supported by the generic Linux ACPI drivers.

This driver used to be named ibm-acpi until kernel 2.6.21 and release 0.13-20070314. It used to be in the drivers/acpi tree, but it was moved to the drivers/misc tree and renamed to thinkpad-acpi for kernel 2.6.22, and release 0.14. It was moved to drivers/platform/x86 for kernel 2.6.29 and release 0.22.

The driver is named "thinkpad-acpi". In some places, like module names and log messages, "thinkpad_acpi" is used because of userspace issues.

"tpacpi" is used as a shorthand where "thinkpad-acpi" would be too long due to length limitations on some Linux kernel versions.

Status

The features currently supported are the following (see below for detailed description):

- Fn key combinations
- Bluetooth enable and disable
- video output switching, expansion control
- ThinkLight on and off
- CMOS/UCMS control
- LED control
- ACPI sounds
- temperature sensors
- Experimental: embedded controller register dump
- LCD brightness control
- Volume control
- Fan control and monitoring: fan speed, fan enable/disable
- WAN enable and disable
- UWB enable and disable

A compatibility table by model and feature is maintained on the web site, http://ibm-acpi.sf.net/. I appreciate any success or failure reports, especially if they add to or correct the compatibility table. Please include the following information in your report:

- ThinkPad model name
- a copy of your ACPI tables, using the "acpidump" utility

- a copy of the output of dmidecode, with serial numbers
  and UUIDs masked off
- which driver features work and which don't
- the observed behavior of non-working features

Any other comments or patches are also more than welcome.

Installation

If you are compiling this driver as included in the Linux kernel
sources, look for the CONFIG_THINKPAD_ACPI Kconfig option.
It is located on the menu path: "Device Drivers" -> "X86 Platform
Specific Device Drivers" -> "ThinkPad ACPI Laptop Extras".

Features

The driver exports two different interfaces to userspace, which can be
used to access the features it provides. One is a legacy procfs-based
interface, which will be removed at some time in the future. The other
is a new sysfs-based interface which is not complete yet.

The procfs interface creates the /proc/acpi/ibm directory. There is a
file under that directory for each feature it supports. The procfs
interface is mostly frozen, and will change very little if at all: it
will not be extended to add any new functionality in the driver, instead
all new functionality will be implemented on the sysfs interface.

The sysfs interface tries to blend in the generic Linux sysfs subsystems
and classes as much as possible. Since some of these subsystems are not
yet ready or stabilized, it is expected that this interface will change,
and any and all userspace programs must deal with it.

Notes about the sysfs interface:

Unlike what was done with the procfs interface, correctness when talking
to the sysfs interfaces will be enforced, as will correctness in the
thinkpad-acpi's implementation of sysfs interfaces.

Also, any bugs in the thinkpad-acpi sysfs driver code or in the
thinkpad-acpi's implementation of the sysfs interfaces will be fixed for
maximum correctness, even if that means changing an interface in
non-compatible ways. As these interfaces mature both in the kernel and
in thinkpad-acpi, such changes should become quite rare.

Applications interfacing to the thinkpad-acpi sysfs interfaces must
follow all sysfs guidelines and correctly process all errors (the sysfs
interface makes extensive use of errors). File descriptors and open /
close operations to the sysfs inodes must also be properly implemented.

The version of thinkpad-acpi's sysfs interface is exported by the driver
as a driver attribute (see below).

Sysfs driver attributes are on the driver's sysfs attribute space,
for 2.6.23+ this is /sys/bus/platform/drivers/thinkpad_acpi/ and
/sys/bus/platform/drivers/thinkpad_hwmon/

Sysfs device attributes are on the thinkpad_acpi device sysfs attribute
space, for 2.6.23+ this is /sys/devices/platform/thinkpad_acpi/.

Sysfs device attributes for the sensors and fan are on the
thinkpad_hwmon device's sysfs attribute space, but you should locate it
looking for a hwmon device with the name attribute of "thinkpad", or
better yet, through libsensors.

Driver version
----

procfs: /proc/acpi/ibm/driver
sysfs driver attribute: version

The driver name and version. No commands can be written to this file.

Sysfs interface version
----

sysfs driver attribute: interface_version

Version of the thinkpad-acpi sysfs interface, as an unsigned long
(output in hex format: 0xAAAABBCC), where:
        AAAA - major revision
        BB - minor revision
        CC - bugfix revision

The sysfs interface version changelog for the driver can be found at the
end of this document. Changes to the sysfs interface done by the kernel
subsystems are not documented here, nor are they tracked by this
attribute.

Changes to the thinkpad-acpi sysfs interface are only considered
non-experimental when they are submitted to Linux mainline, at which
point the changes in this interface are documented and interface_version
may be updated. If you are using any thinkpad-acpi features not yet
sent to mainline for merging, you do so on your own risk: these features
may disappear, or be implemented in a different and incompatible way by
the time they are merged in Linux mainline.

Changes that are backwards-compatible by nature (e.g. the addition of
attributes that do not change the way the other attributes work) do not
always warrant an update of interface_version. Therefore, one must
expect that an attribute might not be there, and deal with it properly
(an attribute not being there *is* a valid way to make it clear that a
feature is not available in sysfs).

Hot keys

---

procfs: /proc/acpi/ibm/hotkey
sysfs device attribute: hotkey_*

In a ThinkPad, the ACPI HKEY handler is responsible for communicating some important events and also keyboard hot key presses to the operating system. Enabling the hotkey functionality of thinkpad-acpi signals the firmware that such a driver is present, and modifies how the ThinkPad firmware will behave in many situations.

The driver enables the HKEY ("hot key") event reporting automatically when loaded, and disables it when it is removed.

The driver will report HKEY events in the following format:

```
ibm/hotkey HKEY 00000080 0000xxxx
```

Some of these events refer to hot key presses, but not all of them.

The driver will generate events over the input layer for hot keys and radio switches, and over the ACPI netlink layer for other events. The input layer support accepts the standard IOCTLs to remap the keycodes assigned to each hot key.

The hot key bit mask allows some control over which hot keys generate events. If a key is "masked" (bit set to 0 in the mask), the firmware will handle it. If it is "unmasked", it signals the firmware that thinkpad-acpi would prefer to handle it, if the firmware would be so kind to allow it (and it often doesn't!).

Not all bits in the mask can be modified. Not all bits that can be modified do anything. Not all hot keys can be individually controlled by the mask. Some models do not support the mask at all. The behaviour of the mask is, therefore, highly dependent on the ThinkPad model.

The driver will filter out any unmasked hotkeys, so even if the firmware doesn't allow disabling an specific hotkey, the driver will not report events for unmasked hotkeys.

Note that unmasking some keys prevents their default behavior. For example, if Fn+F5 is unmasked, that key will no longer enable/disable Bluetooth by itself in firmware.

Note also that not all Fn key combinations are supported through ACPI depending on the ThinkPad model and firmware version. On those ThinkPads, it is still possible to support some extra hotkeys by polling the "CMOS NVRAM" at least 10 times per second. The driver attempts to enables this functionality automatically when required.

procfs notes:

The following commands can be written to the /proc/acpi/ibm/hotkey file:

```
echo 0xffffffff > /proc/acpi/ibm/hotkey -- enable all hot keys
echo 0 > /proc/acpi/ibm/hotkey -- disable all possible hot keys
```

```
... any other 8-hex-digit mask ...
echo reset > /proc/acpi/ibm/hotkey -- restore the recommended mask
```

The following commands have been deprecated and will cause the kernel
to log a warning:

```
echo enable > /proc/acpi/ibm/hotkey -- does nothing
echo disable > /proc/acpi/ibm/hotkey -- returns an error
```

The procfs interface does not support NVRAM polling control. So as to
maintain maximum bug-to-bug compatibility, it does not report any masks,
nor does it allow one to manipulate the hot key mask when the firmware
does not support masks at all, even if NVRAM polling is in use.

sysfs notes:

hotkey_bios_enabled:
DEPRECATED, WILL BE REMOVED SOON.

Returns 0.

hotkey_bios_mask:
DEPRECATED, DON'T USE, WILL BE REMOVED IN THE FUTURE.

Returns the hot keys mask when thinkpad-acpi was loaded.
Upon module unload, the hot keys mask will be restored
to this value. This is always 0x80c, because those are
the hotkeys that were supported by ancient firmware
without mask support.

hotkey_enable:
DEPRECATED, WILL BE REMOVED SOON.

0: returns -EPERM
1: does nothing

hotkey_mask:
bit mask to enable reporting (and depending on
the firmware, ACPI event generation) for each hot key
(see above). Returns the current status of the hot keys
mask, and allows one to modify it.

hotkey_all_mask:
bit mask that should enable event reporting for all
supported hot keys, when echoed to hotkey_mask above.
Unless you know which events need to be handled
passively (because the firmware *will* handle them
anyway), do *not* use hotkey_all_mask. Use
hotkey_recommended_mask, instead. You have been warned.

hotkey_recommended_mask:
bit mask that should enable event reporting for all
supported hot keys, except those which are always
handled by the firmware anyway. Echo it to
hotkey_mask above, to use. This is the default mask
used by the driver.

hotkey_source_mask:
bit mask that selects which hot keys will the driver poll the NVRAM for. This is auto-detected by the driver based on the capabilities reported by the ACPI firmware, but it can be overridden at runtime.

Hot keys whose bits are set in hotkey_source_mask are polled for in NVRAM, and reported as hotkey events if enabled in hotkey_mask. Only a few hot keys are available through CMOS NVRAM polling.

Warning: when in NVRAM mode, the volume up/down/mute keys are synthesized according to changes in the mixer, which uses a single volume up or volume down hotkey press to unmute, as per the ThinkPad volume mixer user interface. When in ACPI event mode, volume up/down/mute events are reported by the firmware and can behave differently (and that behaviour changes with firmware version -- not just with firmware models -- as well as OSI(Linux) state).

hotkey_poll_freq:
frequency in Hz for hot key polling. It must be between 0 and 25 Hz. Polling is only carried out when strictly needed.

Setting hotkey_poll_freq to zero disables polling, and will cause hot key presses that require NVRAM polling to never be reported.

Setting hotkey_poll_freq too low may cause repeated pressings of the same hot key to be misreported as a single key press, or to not even be detected at all. The recommended polling frequency is 10Hz.

hotkey_radio_sw:
If the ThinkPad has a hardware radio switch, this attribute will read 0 if the switch is in the "radios disabled" position, and 1 if the switch is in the "radios enabled" position.

This attribute has poll()/select() support.

hotkey_tablet_mode:
If the ThinkPad has tablet capabilities, this attribute will read 0 if the ThinkPad is in normal mode, and 1 if the ThinkPad is in tablet mode.

This attribute has poll()/select() support.

hotkey_report_mode:
Returns the state of the procfs ACPI event report mode filter for hot keys. If it is set to 1 (the default), all hot key presses are reported both through the input layer and also as ACPI events through procfs (but not

through netlink). If it is set to 2, hot key presses are reported only through the input layer.

This attribute is read-only in kernels 2.6.23 or later, and read-write on earlier kernels.

May return -EPERM (write access locked out by module parameter) or -EACCES (read-only).

wakeup_reason:
Set to 1 if the system is waking up because the user requested a bay ejection. Set to 2 if the system is waking up because the user requested the system to undock. Set to zero for normal wake-ups or wake-ups due to unknown reasons.

This attribute has poll()/select() support.

wakeup_hotunplug_complete:
Set to 1 if the system was waken up because of an undock or bay ejection request, and that request was successfully completed. At this point, it might be useful to send the system back to sleep, at the user's choice. Refer to HKEY events 0x4003 and 0x3003, below.

This attribute has poll()/select() support.

input layer notes:

A Hot key is mapped to a single input layer EV_KEY event, possibly followed by an EV_MSC MSC_SCAN event that shall contain that key's scan code. An EV_SYN event will always be generated to mark the end of the event block.

Do not use the EV_MSC MSC_SCAN events to process keys. They are to be used as a helper to remap keys, only. They are particularly useful when remapping KEY_UNKNOWN keys.

The events are available in an input device, with the following id:

Bus:	BUS_HOST
vendor:	0x1014 (PCI_VENDOR_ID_IBM) or 0x17aa (PCI_VENDOR_ID_LENOVO)
product:	0x5054 ("TP")
version:	0x4101

The version will have its LSB incremented if the keymap changes in a backwards-compatible way. The MSB shall always be 0x41 for this input device. If the MSB is not 0x41, do not use the device as described in this section, as it is either something else (e.g. another input device exported by a thinkpad driver, such as HDAPS) or its functionality has been changed in a non-backwards compatible way.

Adding other event types for other functionalities shall be considered a backwards-compatible change for this input device.

Thinkpad-acpi Hot Key event map (version 0x4101):

ACPI event	Scan code	Key	Notes
0x1001	0x00	FN+F1	-
0x1002	0x01	FN+F2	IBM: battery (rare) Lenovo: Screen lock
0x1003	0x02	FN+F3	Many IBM models always report this hot key, even with hot keys disabled or with Fn+F3 masked off IBM: screen lock, often turns off the ThinkLight as side-effect Lenovo: battery
0x1004	0x03	FN+F4	Sleep button (ACPI sleep button semantics, i.e. sleep-to-RAM). It is always generate some kind of event, either the hot key event or a ACPI sleep button event. The firmware may refuse to generate further FN+F4 key presses until a S3 or S4 ACPI sleep cycle is performed or some time passes.
0x1005	0x04	FN+F5	Radio. Enables/disables the internal Bluetooth hardware and W-WAN card if left in control of the firmware. Does not affect the WLAN card. Should be used to turn on/off all radios (Bluetooth+W-WAN+WLAN), really.
0x1006	0x05	FN+F6	-
0x1007	0x06	FN+F7	Video output cycle. Do you feel lucky today?
0x1008	0x07	FN+F8	IBM: toggle screen expand Lenovo: configure UltraNav, or toggle screen expand
0x1009	0x08	FN+F9	-
..	..	..	..
0x100B	0x0A	FN+F11	-
0x100C	0x0B	FN+F12	Sleep to disk. You are always supposed to handle it yourself, either through the ACPI event, or through a hotkey event.

			The firmware may refuse to generate further FN+F12 key press events until a S3 or S4 ACPI sleep cycle is performed, or some time passes.
0x100D	0x0C	FN+BACKSPACE	-
0x100E	0x0D	FN+INSERT	-
0x100F	0x0E	FN+DELETE	-
0x1010	0x0F	FN+HOME	Brightness up. This key is always handled by the firmware in IBM ThinkPads, even when unmasked. Just leave it alone. For Lenovo ThinkPads with a new BIOS, it has to be handled either by the ACPI OSI, or by userspace. The driver does the right thing, never mess with this.
0x1011	0x10	FN+END	Brightness down. See brightness up for details.
0x1012	0x11	FN+PGUP	ThinkLight toggle. This key is always handled by the firmware, even when unmasked.
0x1013	0x12	FN+PGDOWN	-
0x1014	0x13	FN+SPACE	Zoom key
0x1015	0x14	VOLUME UP	Internal mixer volume up. This key is always handled by the firmware, even when unmasked. NOTE: Lenovo seems to be changing this.
0x1016	0x15	VOLUME DOWN	Internal mixer volume up. This key is always handled by the firmware, even when unmasked. NOTE: Lenovo seems to be changing this.
0x1017	0x16	MUTE	Mute internal mixer. This key is always handled by the firmware, even when unmasked.
0x1018	0x17	THINKPAD	ThinkPad/Access IBM/Lenovo key
0x1019	0x18	unknown	
..	..	..	
0x1020	0x1F	unknown	

The ThinkPad firmware does not allow one to differentiate when most hot keys are pressed or released (either that, or we don't know how to, yet). For these keys, the driver generates a set of events for a key press and immediately issues the same set of events for a key release. It is unknown by the driver if the ThinkPad firmware triggered these events on hot key press or release, but the firmware will do it for either one, not

both.

If a key is mapped to KEY_RESERVED, it generates no input events at all.
If a key is mapped to KEY_UNKNOWN, it generates an input event that
includes an scan code. If a key is mapped to anything else, it will
generate input device EV_KEY events.

In addition to the EV_KEY events, thinkpad-acpi may also issue EV_SW
events for switches:

SW_RFKILL_ALL	T60 and later hardware rfkill rocker switch
SW_TABLET_MODE	Tablet ThinkPads HKEY events 0x5009 and 0x500A

Non hotkey ACPI HKEY event map:

Events that are not propagated by the driver, except for legacy
compatibility purposes when hotkey_report_mode is set to 1:

0x5001	Lid closed
0x5002	Lid opened
0x5009	Tablet swivel: switched to tablet mode
0x500A	Tablet swivel: switched to normal mode
0x7000	Radio Switch may have changed state

Events that are never propagated by the driver:

0x2304	System is waking up from suspend to undock
0x2305	System is waking up from suspend to eject bay
0x2404	System is waking up from hibernation to undock
0x2405	System is waking up from hibernation to eject bay
0x5010	Brightness level changed/control event

Events that are propagated by the driver to userspace:

0x2313	ALARM: System is waking up from suspend because the battery is nearly empty
0x2413	ALARM: System is waking up from hibernation because the battery is nearly empty
0x3003	Bay ejection (see 0x2x05) complete, can sleep again
0x3006	Bay hotplug request (hint to power up SATA link when the optical drive tray is ejected)
0x4003	Undocked (see 0x2x04), can sleep again
0x500B	Tablet pen inserted into its storage bay
0x500C	Tablet pen removed from its storage bay
0x6011	ALARM: battery is too hot
0x6012	ALARM: battery is extremely hot
0x6021	ALARM: a sensor is too hot
0x6022	ALARM: a sensor is extremely hot
0x6030	System thermal table changed

Battery nearly empty alarms are a last resort attempt to get the
operating system to hibernate or shutdown cleanly (0x2313), or shutdown
cleanly (0x2413) before power is lost. They must be acted upon, as the
wake up caused by the firmware will have negated most safety nets...

When any of the "too hot" alarms happen, according to Lenovo the user should suspend or hibernate the laptop (and in the case of battery alarms, unplug the AC adapter) to let it cool down. These alarms do signal that something is wrong, they should never happen on normal operating conditions.

The "extremely hot" alarms are emergencies. According to Lenovo, the operating system is to force either an immediate suspend or hibernate cycle, or a system shutdown. Obviously, something is very wrong if this happens.

Compatibility notes:

ibm-acpi and thinkpad-acpi 0.15 (mainline kernels before 2.6.23) never supported the input layer, and sent events over the procfs ACPI event interface.

To avoid sending duplicate events over the input layer and the ACPI event interface, thinkpad-acpi 0.16 implements a module parameter (hotkey_report_mode), and also a sysfs device attribute with the same name.

Make no mistake here: userspace is expected to switch to using the input layer interface of thinkpad-acpi, together with the ACPI netlink event interface in kernels 2.6.23 and later, or with the ACPI procfs event interface in kernels 2.6.22 and earlier.

If no hotkey_report_mode module parameter is specified (or it is set to zero), the driver defaults to mode 1 (see below), and on kernels 2.6.22 and earlier, also allows one to change the hotkey_report_mode through sysfs. In kernels 2.6.23 and later, where the netlink ACPI event interface is available, hotkey_report_mode cannot be changed through sysfs (it is read-only).

If the hotkey_report_mode module parameter is set to 1 or 2, it cannot be changed later through sysfs (any writes will return -EPERM to signal that hotkey_report_mode was locked. On 2.6.23 and later, where hotkey_report_mode cannot be changed at all, writes will return -EACCES).

hotkey_report_mode set to 1 makes the driver export through the procfs ACPI event interface all hot key presses (which are *also* sent to the input layer). This is a legacy compatibility behaviour, and it is also the default mode of operation for the driver.

hotkey_report_mode set to 2 makes the driver filter out the hot key presses from the procfs ACPI event interface, so these events will only be sent through the input layer. Userspace that has been updated to use the thinkpad-acpi input layer interface should set hotkey_report_mode to 2.

Hot key press events are never sent to the ACPI netlink event interface. Really up-to-date userspace under kernel 2.6.23 and later is to use the netlink interface and the input layer interface, and don't bother at all with hotkey_report_mode.

Brightness hotkey notes:

Don't mess with the brightness hotkeys in a Thinkpad. If you want notifications for OSD, use the sysfs backlight class event support.

The driver will issue KEY_BRIGHTNESS_UP and KEY_BRIGHTNESS_DOWN events automatically for the cases were userspace has to do something to implement brightness changes. When you override these events, you will either fail to handle properly the ThinkPads that require explicit action to change backlight brightness, or the ThinkPads that require that no action be taken to work properly.

Bluetooth
----

procfs: /proc/acpi/ibm/bluetooth
sysfs device attribute: bluetooth_enable (deprecated)
sysfs rfkill class: switch "tpacpi_bluetooth_sw"

This feature shows the presence and current state of a ThinkPad Bluetooth device in the internal ThinkPad CDC slot.

If the ThinkPad supports it, the Bluetooth state is stored in NVRAM, so it is kept across reboots and power-off.

Procfs notes:

If Bluetooth is installed, the following commands can be used:

```
echo enable > /proc/acpi/ibm/bluetooth
echo disable > /proc/acpi/ibm/bluetooth
```

Sysfs notes:

If the Bluetooth CDC card is installed, it can be enabled / disabled through the "bluetooth_enable" thinkpad-acpi device attribute, and its current status can also be queried.

enable:
0: disables Bluetooth / Bluetooth is disabled
1: enables Bluetooth / Bluetooth is enabled.

Note: this interface has been superseded by the generic rfkill class. It has been deprecated, and it will be removed in year 2010.

rfkill controller switch "tpacpi_bluetooth_sw": refer to Documentation/rfkill.txt for details.

Video output control -- /proc/acpi/ibm/video

This feature allows control over the devices used for video output - LCD, CRT or DVI (if available). The following commands are available:

```
echo lcd_enable > /proc/acpi/ibm/video
echo lcd_disable > /proc/acpi/ibm/video
echo crt_enable > /proc/acpi/ibm/video
echo crt_disable > /proc/acpi/ibm/video
echo dvi_enable > /proc/acpi/ibm/video
echo dvi_disable > /proc/acpi/ibm/video
echo auto_enable > /proc/acpi/ibm/video
echo auto_disable > /proc/acpi/ibm/video
echo expand_toggle > /proc/acpi/ibm/video
echo video_switch > /proc/acpi/ibm/video
```

NOTE: Access to this feature is restricted to processes owning the CAP_SYS_ADMIN capability for safety reasons, as it can interact badly enough with some versions of X.org to crash it.

Each video output device can be enabled or disabled individually. Reading /proc/acpi/ibm/video shows the status of each device.

Automatic video switching can be enabled or disabled. When automatic video switching is enabled, certain events (e.g. opening the lid, docking or undocking) cause the video output device to change automatically. While this can be useful, it also causes flickering and, on the X40, video corruption. By disabling automatic switching, the flickering or video corruption can be avoided.

The video_switch command cycles through the available video outputs (it simulates the behavior of Fn-F7).

Video expansion can be toggled through this feature. This controls whether the display is expanded to fill the entire LCD screen when a mode with less than full resolution is used. Note that the current video expansion status cannot be determined through this feature.

Note that on many models (particularly those using Radeon graphics chips) the X driver configures the video card in a way which prevents Fn-F7 from working. This also disables the video output switching features of this driver, as it uses the same ACPI methods as Fn-F7. Video switching on the console should still work.

UPDATE: refer to https://bugs.freedesktop.org/show_bug.cgi?id=2000

ThinkLight control

procfs: /proc/acpi/ibm/light
sysfs attributes: as per LED class, for the "tpacpi::thinklight" LED

procfs notes:

The ThinkLight status can be read and set through the procfs interface. A few models which do not make the status available will show the ThinkLight status as "unknown". The available commands are:

```
echo on  > /proc/acpi/ibm/light
```

```
        echo off > /proc/acpi/ibm/light
```

sysfs notes:

The ThinkLight sysfs interface is documented by the LED class documentation, in Documentation/leds-class.txt. The ThinkLight LED name is "tpacpi::thinklight".

Due to limitations in the sysfs LED class, if the status of the ThinkLight cannot be read or if it is unknown, thinkpad-acpi will report it as "off". It is impossible to know if the status returned through sysfs is valid.

CMOS/UCMS control

procfs: /proc/acpi/ibm/cmos
sysfs device attribute: cmos_command

This feature is mostly used internally by the ACPI firmware to keep the legacy CMOS NVRAM bits in sync with the current machine state, and to record this state so that the ThinkPad will retain such settings across reboots.

Some of these commands actually perform actions in some ThinkPad models, but this is expected to disappear more and more in newer models. As an example, in a T43 and in a X40, commands 12 and 13 still control the ThinkLight state for real, but commands 0 to 2 don't control the mixer anymore (they have been phased out) and just update the NVRAM.

The range of valid cmos command numbers is 0 to 21, but not all have an effect and the behavior varies from model to model. Here is the behavior on the X40 (tpb is the ThinkPad Buttons utility):

```
        0 - Related to "Volume down" key press
        1 - Related to "Volume up" key press
        2 - Related to "Mute on" key press
        3 - Related to "Access IBM" key press
        4 - Related to "LCD brightness up" key press
        5 - Related to "LCD brightness down" key press
        11 - Related to "toggle screen expansion" key press/function
        12 - Related to "ThinkLight on"
        13 - Related to "ThinkLight off"
        14 - Related to "ThinkLight" key press (toggle ThinkLight)
```

The cmos command interface is prone to firmware split-brain problems, as in newer ThinkPads it is just a compatibility layer. Do not use it, it is exported just as a debug tool.

LED control

procfs: /proc/acpi/ibm/led
sysfs attributes: as per LED class, see below for names

Some of the LED indicators can be controlled through this feature. On

some older ThinkPad models, it is possible to query the status of the LED indicators as well. Newer ThinkPads cannot query the real status of the LED indicators.

Because misuse of the LEDs could induce an unaware user to perform dangerous actions (like undocking or ejecting a bay device while the buses are still active), or mask an important alarm (such as a nearly empty battery, or a broken battery), access to most LEDs is restricted.

Unrestricted access to all LEDs requires that thinkpad-acpi be compiled with the CONFIG_THINKPAD_ACPI_UNSAFE_LEDS option enabled. Distributions must never enable this option. Individual users that are aware of the consequences are welcome to enabling it.

procfs notes:

The available commands are:

```
echo '<LED number> on' >/proc/acpi/ibm/led
echo '<LED number> off' >/proc/acpi/ibm/led
echo '<LED number> blink' >/proc/acpi/ibm/led
```

The <LED number> range is 0 to 15. The set of LEDs that can be controlled varies from model to model. Here is the common ThinkPad mapping:

```
0 - power
1 - battery (orange)
2 - battery (green)
3 - UltraBase/dock
4 - UltraBay
5 - UltraBase battery slot
6 - (unknown)
7 - standby
8 - dock status 1
9 - dock status 2
10, 11 - (unknown)
12 - thinkvantage
13, 14, 15 - (unknown)
```

All of the above can be turned on and off and can be made to blink.

sysfs notes:

The ThinkPad LED sysfs interface is described in detail by the LED class documentation, in Documentation/leds-class.txt.

The LEDs are named (in LED ID order, from 0 to 12):
"tpacpi::power", "tpacpi:orange:batt", "tpacpi:green:batt", "tpacpi::dock_active", "tpacpi::bay_active", "tpacpi::dock_batt", "tpacpi::unknown_led", "tpacpi::standby", "tpacpi::dock_status1", "tpacpi::dock_status2", "tpacpi::unknown_led2", "tpacpi::unknown_led3", "tpacpi::thinkvantage".

Due to limitations in the sysfs LED class, if the status of the LED

indicators cannot be read due to an error, thinkpad-acpi will report it as
a brightness of zero (same as LED off).

If the thinkpad firmware doesn't support reading the current status,
trying to read the current LED brightness will just return whatever
brightness was last written to that attribute.

These LEDs can blink using hardware acceleration. To request that a
ThinkPad indicator LED should blink in hardware accelerated mode, use the
"timer" trigger, and leave the delay_on and delay_off parameters set to
zero (to request hardware acceleration autodetection).

LEDs that are known not to exist in a given ThinkPad model are not
made available through the sysfs interface. If you have a dock and you
notice there are LEDs listed for your ThinkPad that do not exist (and
are not in the dock), or if you notice that there are missing LEDs,
a report to ibm-acpi-devel@lists.sourceforge.net is appreciated.

ACPI sounds -- /proc/acpi/ibm/beep

The BEEP method is used internally by the ACPI firmware to provide
audible alerts in various situations. This feature allows the same
sounds to be triggered manually.

The commands are non-negative integer numbers:

        echo <number> >/proc/acpi/ibm/beep

The valid <number> range is 0 to 17. Not all numbers trigger sounds
and the sounds vary from model to model. Here is the behavior on the
X40:

        0 - stop a sound in progress (but use 17 to stop 16)
        2 - two beeps, pause, third beep ("low battery")
        3 - single beep
        4 - high, followed by low-pitched beep ("unable")
        5 - single beep
        6 - very high, followed by high-pitched beep ("AC/DC")
        7 - high-pitched beep
        9 - three short beeps
        10 - very long beep
        12 - low-pitched beep
        15 - three high-pitched beeps repeating constantly, stop with 0
        16 - one medium-pitched beep repeating constantly, stop with 17
        17 - stop 16

Temperature sensors
----

procfs: /proc/acpi/ibm/thermal
sysfs device attributes: (hwmon "thinkpad") temp*_input

Most ThinkPads include six or more separate temperature sensors but only

expose the CPU temperature through the standard ACPI methods. This
feature shows readings from up to eight different sensors on older
ThinkPads, and up to sixteen different sensors on newer ThinkPads.

For example, on the X40, a typical output may be:
temperatures: 42 42 45 41 36 -128 33 -128

On the T43/p, a typical output may be:
temperatures: 48 48 36 52 38 -128 31 -128 48 52 48 -128 -128 -128 -128 -128

The mapping of thermal sensors to physical locations varies depending on
system-board model (and thus, on ThinkPad model).

http://thinkwiki.org/wiki/Thermal_Sensors is a public wiki page that
tries to track down these locations for various models.

Most (newer?) models seem to follow this pattern:

1: CPU
2: (depends on model)
3: (depends on model)
4: GPU
5: Main battery: main sensor
6: Bay battery: main sensor
7: Main battery: secondary sensor
8: Bay battery: secondary sensor
9-15: (depends on model)

For the R51 (source: Thomas Gruber):
2: Mini-PCI
3: Internal HDD

For the T43, T43/p (source: Shmidoax/Thinkwiki.org)
http://thinkwiki.org/wiki/Thermal_Sensors#ThinkPad_T43.2C_T43p
2: System board, left side (near PCMCIA slot), reported as HDAPS temp
3: PCMCIA slot
9: MCH (northbridge) to DRAM Bus
10: Clock-generator, mini-pci card and ICH (southbridge), under Mini-PCI
    card, under touchpad
11: Power regulator, underside of system board, below F2 key

The A31 has a very atypical layout for the thermal sensors
(source: Milos Popovic, http://thinkwiki.org/wiki/Thermal_Sensors#ThinkPad_A31)
1: CPU
2: Main Battery: main sensor
3: Power Converter
4: Bay Battery: main sensor
5: MCH (northbridge)
6: PCMCIA/ambient
7: Main Battery: secondary sensor
8: Bay Battery: secondary sensor

Procfs notes:
        Readings from sensors that are not available return -128.
        No commands can be written to this file.

Sysfs notes:

Sensors that are not available return the ENXIO error. This status may change at runtime, as there are hotplug thermal sensors, like those inside the batteries and docks.

thinkpad-acpi thermal sensors are reported through the hwmon subsystem, and follow all of the hwmon guidelines at Documentation/hwmon.

EXPERIMENTAL: Embedded controller register dump -- /proc/acpi/ibm/ecdump
---

This feature is marked EXPERIMENTAL because the implementation directly accesses hardware registers and may not work as expected. USE WITH CAUTION! To use this feature, you need to supply the experimental=1 parameter when loading the module.

This feature dumps the values of 256 embedded controller registers. Values which have changed since the last time the registers were dumped are marked with a star:

```
[root@x40 ibm-acpi]# cat /proc/acpi/ibm/ecdump
EC       +00 +01 +02 +03 +04 +05 +06 +07 +08 +09 +0a +0b +0c +0d +0e +0f
EC 0x00:  a7  47  87  01  fe  96  00  08  01  00  cb  00  00  00  40  00
EC 0x10:  00  00  ff  ff  f4  3c  87  09  01  ff  42  01  ff  ff  0d  00
EC 0x20:  00  00  00  00  00  00  00  00  00  00  00  03  43  00  00  80
EC 0x30:  01  07  1a  00  30  04  00  00 *85  00  00  10  00  50  00  00
EC 0x40:  00  00  00  00  00  00  14  01  00  04  00  00  00  00  00  00
EC 0x50:  00  c0  02  0d  00  01  01  02  02  03  03  03  03 *bc *02 *bc
EC 0x60: *02 *bc *02  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0x70:  00  00  00  00  00  12  30  40 *24 *26 *2c *27 *20  80 *1f  80
EC 0x80:  00  00  00  06 *37 *0e  03  00  00  00  0e  07  00  00  00  00
EC 0x90:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0xa0: *ff  09  ff  09  ff  ff *64  00 *00 *00 *a2  41 *ff *ff *e0  00
EC 0xb0:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0xc0:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0xd0:  03  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0xe0:  00  00  00  00  00  00  00  00  11  20  49  04  24  06  55  03
EC 0xf0:  31  55  48  54  35  38  57  57  08  2f  45  73  07  65  6c  1a
```

This feature can be used to determine the register holding the fan speed on some models. To do that, do the following:

- make sure the battery is fully charged
- make sure the fan is running
- run 'cat /proc/acpi/ibm/ecdump' several times, once per second or so

The first step makes sure various charging-related values don't vary. The second ensures that the fan-related values do vary, since the fan speed fluctuates a bit. The third will (hopefully) mark the fan register with a star:

```
[root@x40 ibm-acpi]# cat /proc/acpi/ibm/ecdump
EC       +00 +01 +02 +03 +04 +05 +06 +07 +08 +09 +0a +0b +0c +0d +0e +0f
```

```
EC 0x00:  a7  47  87  01  fe  96  00  08  01  00  cb  00  00  00  40  00
EC 0x10:  00  00  ff  ff  f4  3c  87  09  01  ff  42  01  ff  ff  0d  00
EC 0x20:  00  00  00  00  00  00  00  00  00  00  00  03  43  00  00  80
EC 0x30:  01  07  1a  00  30  04  00  00  85  00  00  10  00  50  00  00
EC 0x40:  00  00  00  00  00  00  14  01  00  04  00  00  00  00  00  00
EC 0x50:  00  c0  02  0d  00  01  01  02  02  03  03  03  03  bc  02  bc
EC 0x60:  02  bc  02  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0x70:  00  00  00  00  00  12  30  40  24  27  2c  27  21  80  1f  80
EC 0x80:  00  00  00  06 *be  0d  03  00  00  00  0e  07  00  00  00  00
EC 0x90:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0xa0:  ff  09  ff  09  ff  ff  64  00  00  00  a2  41  ff  ff  e0  00
EC 0xb0:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0xc0:  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0xd0:  03  00  00  00  00  00  00  00  00  00  00  00  00  00  00  00
EC 0xe0:  00  00  00  00  00  00  00  00  11  20  49  04  24  06  55  03
EC 0xf0:  31  55  48  54  35  38  57  57  08  2f  45  73  07  65  6c  1a
```

Another set of values that varies often is the temperature readings. Since temperatures don't change vary fast, you can take several quick dumps to eliminate them.

You can use a similar method to figure out the meaning of other embedded controller registers - e.g. make sure nothing else changes except the charging or discharging battery to determine which registers contain the current battery capacity, etc. If you experiment with this, do send me your results (including some complete dumps with a description of the conditions when they were taken.)

LCD brightness control

procfs: /proc/acpi/ibm/brightness
sysfs backlight device "thinkpad_screen"

This feature allows software control of the LCD brightness on ThinkPad models which don't have a hardware brightness slider.

It has some limitations: the LCD backlight cannot be actually turned on or off by this interface, it just controls the backlight brightness level.

On IBM (and some of the earlier Lenovo) ThinkPads, the backlight control has eight brightness levels, ranging from 0 to 7. Some of the levels may not be distinct. Later Lenovo models that implement the ACPI display backlight brightness control methods have 16 levels, ranging from 0 to 15.

For IBM ThinkPads, there are two interfaces to the firmware for direct brightness control, EC and UCMS (or CMOS). To select which one should be used, use the brightness_mode module parameter: brightness_mode=1 selects EC mode, brightness_mode=2 selects UCMS mode, brightness_mode=3 selects EC mode with NVRAM backing (so that brightness changes are remembered across shutdown/reboot).

The driver tries to select which interface to use from a table of

defaults for each ThinkPad model. If it makes a wrong choice, please report this as a bug, so that we can fix it.

Lenovo ThinkPads only support brightness_mode=2 (UCMS).

When display backlight brightness controls are available through the standard ACPI interface, it is best to use it instead of this direct ThinkPad-specific interface. The driver will disable its native backlight brightness control interface if it detects that the standard ACPI interface is available in the ThinkPad.

The brightness_enable module parameter can be used to control whether the LCD brightness control feature will be enabled when available. brightness_enable=0 forces it to be disabled. brightness_enable=1 forces it to be enabled when available, even if the standard ACPI interface is also available.

Procfs notes:

The available commands are:

```
echo up   >/proc/acpi/ibm/brightness
echo down >/proc/acpi/ibm/brightness
echo 'level <level>' >/proc/acpi/ibm/brightness
```

Sysfs notes:

The interface is implemented through the backlight sysfs class, which is poorly documented at this time.

Locate the thinkpad_screen device under /sys/class/backlight, and inside it there will be the following attributes:

max_brightness:
: Reads the maximum brightness the hardware can be set to. The minimum is always zero.

actual_brightness:
: Reads what brightness the screen is set to at this instant.

brightness:
: Writes request the driver to change brightness to the given value. Reads will tell you what brightness the driver is trying to set the display to when "power" is set to zero and the display has not been dimmed by a kernel power management event.

power:
: power management mode, where 0 is "display on", and 1 to 3 will dim the display backlight to brightness level 0 because thinkpad-acpi cannot really turn the backlight off. Kernel power management events can temporarily increase the current power management level, i.e. they can dim the display.

WARNING:

    Whatever you do, do NOT ever call thinkpad-acpi backlight-level change
    interface and the ACPI-based backlight level change interface
    (available on newer BIOSes, and driven by the Linux ACPI video driver)
    at the same time. The two will interact in bad ways, do funny things,
    and maybe reduce the life of the backlight lamps by needlessly kicking
    its level up and down at every change.

Volume control (Console Audio control)
----

procfs: /proc/acpi/ibm/volume
ALSA: "ThinkPad Console Audio Control", default ID: "ThinkPadEC"

NOTE: by default, the volume control interface operates in read-only
mode, as it is supposed to be used for on-screen-display purposes.
The read/write mode can be enabled through the use of the
"volume_control=1" module parameter.

NOTE: distros are urged to not enable volume_control by default, this
should be done by the local admin only. The ThinkPad UI is for the
console audio control to be done through the volume keys only, and for
the desktop environment to just provide on-screen-display feedback.
Software volume control should be done only in the main AC97/HDA
mixer.

About the ThinkPad Console Audio control:

ThinkPads have a built-in amplifier and muting circuit that drives the
console headphone and speakers. This circuit is after the main AC97
or HDA mixer in the audio path, and under exclusive control of the
firmware.

ThinkPads have three special hotkeys to interact with the console
audio control: volume up, volume down and mute.

It is worth noting that the normal way the mute function works (on
ThinkPads that do not have a "mute LED") is:

1. Press mute to mute. It will *always* mute, you can press it as
   many times as you want, and the sound will remain mute.

2. Press either volume key to unmute the ThinkPad (it will _not_
   change the volume, it will just unmute).

This is a very superior design when compared to the cheap software-only
mute-toggle solution found on normal consumer laptops: you can be
absolutely sure the ThinkPad will not make noise if you press the mute
button, no matter the previous state.

The IBM ThinkPads, and the earlier Lenovo ThinkPads have variable-gain
amplifiers driving the speakers and headphone output, and the firmware
also handles volume control for the headphone and speakers on these

ThinkPads without any help from the operating system (this volume control stage exists after the main AC97 or HDA mixer in the audio path).

The newer Lenovo models only have firmware mute control, and depend on the main HDA mixer to do volume control (which is done by the operating system). In this case, the volume keys are filtered out for unmute key press (there are some firmware bugs in this area) and delivered as normal key presses to the operating system (thinkpad-acpi is not involved).

The ThinkPad-ACPI volume control:

The preferred way to interact with the Console Audio control is the ALSA interface.

The legacy procfs interface allows one to read the current state, and if volume control is enabled, accepts the following commands:

```
echo up   >/proc/acpi/ibm/volume
echo down >/proc/acpi/ibm/volume
echo mute >/proc/acpi/ibm/volume
echo unmute >/proc/acpi/ibm/volume
echo 'level <level>' >/proc/acpi/ibm/volume
```

The <level> number range is 0 to 14 although not all of them may be distinct. To unmute the volume after the mute command, use either the up or down command (the level command will not unmute the volume), or the unmute command.

You can use the volume_capabilities parameter to tell the driver whether your thinkpad has volume control or mute-only control: volume_capabilities=1 for mixers with mute and volume control, volume_capabilities=2 for mixers with only mute control.

If the driver misdetects the capabilities for your ThinkPad model, please report this to ibm-acpi-devel@lists.sourceforge.net, so that we can update the driver.

There are two strategies for volume control. To select which one should be used, use the volume_mode module parameter: volume_mode=1 selects EC mode, and volume_mode=3 selects EC mode with NVRAM backing (so that volume/mute changes are remembered across shutdown/reboot).

The driver will operate in volume_mode=3 by default. If that does not work well on your ThinkPad model, please report this to ibm-acpi-devel@lists.sourceforge.net.

The driver supports the standard ALSA module parameters. If the ALSA mixer is disabled, the driver will disable all volume functionality.

Fan control and monitoring: fan speed, fan enable/disable
---

procfs: /proc/acpi/ibm/fan
sysfs device attributes: (hwmon "thinkpad") fan1_input, pwm1,
                         pwm1_enable, fan2_input
sysfs hwmon driver attributes: fan_watchdog

NOTE NOTE NOTE: fan control operations are disabled by default for safety reasons. To enable them, the module parameter "fan_control=1" must be given to thinkpad-acpi.

This feature attempts to show the current fan speed, control mode and other fan data that might be available. The speed is read directly from the hardware registers of the embedded controller. This is known to work on later R, T, X and Z series ThinkPads but may show a bogus value on other models.

Some Lenovo ThinkPads support a secondary fan. This fan cannot be controlled separately, it shares the main fan control.

Fan levels:

Most ThinkPad fans work in "levels" at the firmware interface. Level 0 stops the fan. The higher the level, the higher the fan speed, although adjacent levels often map to the same fan speed. 7 is the highest level, where the fan reaches the maximum recommended speed.

Level "auto" means the EC changes the fan level according to some internal algorithm, usually based on readings from the thermal sensors.

There is also a "full-speed" level, also known as "disengaged" level. In this level, the EC disables the speed-locked closed-loop fan control, and drives the fan as fast as it can go, which might exceed hardware limits, so use this level with caution.

The fan usually ramps up or down slowly from one speed to another, and it is normal for the EC to take several seconds to react to fan commands. The full-speed level may take up to two minutes to ramp up to maximum speed, and in some ThinkPads, the tachometer readings go stale while the EC is transitioning to the full-speed level.

WARNING WARNING WARNING: do not leave the fan disabled unless you are monitoring all of the temperature sensor readings and you are ready to enable it if necessary to avoid overheating.

An enabled fan in level "auto" may stop spinning if the EC decides the ThinkPad is cool enough and doesn't need the extra airflow. This is normal, and the EC will spin the fan up if the various thermal readings rise too much.

On the X40, this seems to depend on the CPU and HDD temperatures. Specifically, the fan is turned on when either the CPU temperature climbs to 56 degrees or the HDD temperature climbs to 46 degrees. The fan is turned off when the CPU temperature drops to 49 degrees and the HDD temperature drops to 41 degrees. These thresholds cannot currently be controlled.

The ThinkPad's ACPI DSDT code will reprogram the fan on its own when

certain conditions are met. It will override any fan programming done through thinkpad-acpi.

The thinkpad-acpi kernel driver can be programmed to revert the fan level to a safe setting if userspace does not issue one of the procfs fan commands: "enable", "disable", "level" or "watchdog", or if there are no writes to pwm1_enable (or to pwm1 *if and only if* pwm1_enable is set to 1, manual mode) within a configurable amount of time of up to 120 seconds. This functionality is called fan safety watchdog.

Note that the watchdog timer stops after it enables the fan. It will be rearmed again automatically (using the same interval) when one of the above mentioned fan commands is received. The fan watchdog is, therefore, not suitable to protect against fan mode changes made through means other than the "enable", "disable", and "level" procfs fan commands, or the hwmon fan control sysfs interface.

Procfs notes:

The fan may be enabled or disabled with the following commands:

```
echo enable  >/proc/acpi/ibm/fan
echo disable >/proc/acpi/ibm/fan
```

Placing a fan on level 0 is the same as disabling it. Enabling a fan will try to place it in a safe level if it is too slow or disabled.

The fan level can be controlled with the command:

```
echo 'level <level>' > /proc/acpi/ibm/fan
```

Where <level> is an integer from 0 to 7, or one of the words "auto" or "full-speed" (without the quotes). Not all ThinkPads support the "auto" and "full-speed" levels. The driver accepts "disengaged" as an alias for "full-speed", and reports it as "disengaged" for backwards compatibility.

On the X31 and X40 (and ONLY on those models), the fan speed can be controlled to a certain degree. Once the fan is running, it can be forced to run faster or slower with the following command:

```
echo 'speed <speed>' > /proc/acpi/ibm/fan
```

The sustainable range of fan speeds on the X40 appears to be from about 3700 to about 7350. Values outside this range either do not have any effect or the fan speed eventually settles somewhere in that range. The fan cannot be stopped or started with this command. This functionality is incomplete, and not available through the sysfs interface.

To program the safety watchdog, use the "watchdog" command.

```
echo 'watchdog <interval in seconds>' > /proc/acpi/ibm/fan
```

If you want to disable the watchdog, use 0 as the interval.

Sysfs notes:

The sysfs interface follows the hwmon subsystem guidelines for the most part, and the exception is the fan safety watchdog.

Writes to any of the sysfs attributes may return the EINVAL error if that operation is not supported in a given ThinkPad or if the parameter is out-of-bounds, and EPERM if it is forbidden. They may also return EINTR (interrupted system call), and EIO (I/O error while trying to talk to the firmware).

Features not yet implemented by the driver return ENOSYS.

hwmon device attribute pwm1_enable:
	0: PWM offline (fan is set to full-speed mode)
	1: Manual PWM control (use pwm1 to set fan level)
	2: Hardware PWM control (EC "auto" mode)
	3: reserved (Software PWM control, not implemented yet)

	Modes 0 and 2 are not supported by all ThinkPads, and the
	driver is not always able to detect this. If it does know a
	mode is unsupported, it will return -EINVAL.

hwmon device attribute pwm1:
	Fan level, scaled from the firmware values of 0-7 to the hwmon
	scale of 0-255. 0 means fan stopped, 255 means highest normal
	speed (level 7).

	This attribute only commands the fan if pmw1_enable is set to 1
	(manual PWM control).

hwmon device attribute fan1_input:
	Fan tachometer reading, in RPM. May go stale on certain
	ThinkPads while the EC transitions the PWM to offline mode,
	which can take up to two minutes. May return rubbish on older
	ThinkPads.

hwmon device attribute fan2_input:
	Fan tachometer reading, in RPM, for the secondary fan.
	Available only on some ThinkPads. If the secondary fan is
	not installed, will always read 0.

hwmon driver attribute fan_watchdog:
	Fan safety watchdog timer interval, in seconds. Minimum is
	1 second, maximum is 120 seconds. 0 disables the watchdog.

To stop the fan: set pwm1 to zero, and pwm1_enable to 1.

To start the fan in a safe mode: set pwm1_enable to 2. If that fails with EINVAL, try to set pwm1_enable to 1 and pwm1 to at least 128 (255 would be the safest choice, though).

WAN
---

procfs: /proc/acpi/ibm/wan

sysfs device attribute: wwan_enable (deprecated)
sysfs rfkill class: switch "tpacpi_wwan_sw"

This feature shows the presence and current state of the built-in
Wireless WAN device.

If the ThinkPad supports it, the WWAN state is stored in NVRAM,
so it is kept across reboots and power-off.

It was tested on a Lenovo ThinkPad X60. It should probably work on other
ThinkPad models which come with this module installed.

Procfs notes:

If the W-WAN card is installed, the following commands can be used:

        echo enable > /proc/acpi/ibm/wan
        echo disable > /proc/acpi/ibm/wan

Sysfs notes:

        If the W-WAN card is installed, it can be enabled /
        disabled through the "wwan_enable" thinkpad-acpi device
        attribute, and its current status can also be queried.

        enable:
                0: disables WWAN card / WWAN card is disabled
                1: enables WWAN card / WWAN card is enabled.

        Note: this interface has been superseded by the generic rfkill
        class. It has been deprecated, and it will be removed in year
        2010.

        rfkill controller switch "tpacpi_wwan_sw": refer to
        Documentation/rfkill.txt for details.

EXPERIMENTAL: UWB

This feature is marked EXPERIMENTAL because it has not been extensively
tested and validated in various ThinkPad models yet. The feature may not
work as expected. USE WITH CAUTION! To use this feature, you need to supply
the experimental=1 parameter when loading the module.

sysfs rfkill class: switch "tpacpi_uwb_sw"

This feature exports an rfkill controller for the UWB device, if one is
present and enabled in the BIOS.

Sysfs notes:

        rfkill controller switch "tpacpi_uwb_sw": refer to
        Documentation/rfkill.txt for details.

Multiple Commands, Module Parameters

Multiple commands can be written to the proc files in one shot by
separating them with commas, for example:

```
echo enable,0xffff > /proc/acpi/ibm/hotkey
echo lcd_disable,crt_enable > /proc/acpi/ibm/video
```

Commands can also be specified when loading the thinkpad-acpi module,
for example:

```
modprobe thinkpad_acpi hotkey=enable,0xffff video=auto_disable
```

Enabling debugging output

The module takes a debug parameter which can be used to selectively
enable various classes of debugging output, for example:

```
modprobe thinkpad_acpi debug=0xffff
```

will enable all debugging output classes. It takes a bitmask, so
to enable more than one output class, just add their values.

Debug bitmask	Description
0x8000	Disclose PID of userspace programs accessing some functions of the driver
0x0001	Initialization and probing
0x0002	Removal
0x0004	RF Transmitter control (RFKILL) (bluetooth, WWAN, UWB...)
0x0008	HKEY event interface, hotkeys
0x0010	Fan control
0x0020	Backlight brightness
0x0040	Audio mixer/volume control

There is also a kernel build option to enable more debugging
information, which may be necessary to debug driver problems.

The level of debugging information output by the driver can be changed
at runtime through sysfs, using the driver attribute debug_level. The
attribute takes the same bitmask as the debug module parameter above.

Force loading of module

If thinkpad-acpi refuses to detect your ThinkPad, you can try to specify
the module parameter force_load=1. Regardless of whether this works or
not, please contact ibm-acpi-devel@lists.sourceforge.net with a report.

Sysfs interface changelog:

0x000100: Initial sysfs support, as a single platform driver and device.

0x000200: Hot key support for 32 hot keys, and radio slider switch support.

0x010000: Hot keys are now handled by default over the input layer, the radio switch generates input event EV_RADIO, and the driver enables hot key handling by default in the firmware.

0x020000: ABI fix: added a separate hwmon platform device and driver, which must be located by name (thinkpad) and the hwmon class for libsensors4 (lm-sensors 3) compatibility. Moved all hwmon attributes to this new platform device.

0x020100: Marker for thinkpad-acpi with hot key NVRAM polling support. If you must, use it to know you should not start a userspace NVRAM poller (allows to detect when NVRAM is compiled out by the user because it is unneeded/undesired in the first place).

0x020101: Marker for thinkpad-acpi with hot key NVRAM polling and proper hotkey_mask semantics (version 8 of the NVRAM polling patch). Some development snapshots of 0.18 had an earlier version that did strange things to hotkey_mask.

0x020200: Add poll()/select() support to the following attributes: hotkey_radio_sw, wakeup_hotunplug_complete, wakeup_reason

0x020300: hotkey enable/disable support removed, attributes hotkey_bios_enabled and hotkey_enable deprecated and marked for removal.

0x020400: Marker for 16 LEDs support. Also, LEDs that are known to not exist in a given model are not registered with the LED sysfs class anymore.

0x020500: Updated hotkey driver, hotkey_mask is always available and it is always able to disable hot keys. Very old thinkpads are properly supported. hotkey_bios_mask is deprecated and marked for removal.

0x020600: Marker for backlight change event support.

0x020700: Support for mute-only mixers.
Volume control in read-only mode by default.
Marker for ALSA mixer support.