NAME

acpi - Advanced Configuration and Power Management support

SYNOPSIS

device acpi

options ACPI_DEBUG
options DDB

DESCRIPTION

The **acpi** driver provides support for the Intel/Microsoft/Compaq/Toshiba ACPI standard. This support includes platform hardware discovery (superseding the PnP and PCI BIOS), as well as power management (superseding APM) and other features. ACPI core support is provided by the ACPI CA reference implementation from Intel.

Note that the **acpi** driver is automatically loaded by the loader(8), and should only be compiled into the kernel on platforms where ACPI is mandatory.

SYSCTL VARIABLES

The **acpi** driver is intended to provide power management without user intervention. If the default settings are not optimal, the following sysctls can be used to modify or monitor **acpi** behavior. Note that some variables will be available only if the given hardware supports them (such as *hw.acpi.acline*).

debug.acpi.enable_debug_objects

Enable dumping Debug objects without **options ACPI_DEBUG**. Default is 0, ignore Debug objects.

dev.cpu.N.cx_usage

Debugging information listing the percent of total usage for each sleep state. The values are reset when *dev.cpu.N.cx_lowest* is modified.

dev.cpu.N.cx_lowest

Lowest Cx state to use for idling the CPU. A scheduling algorithm will select states between C1 and this setting as system load dictates. To enable ACPI CPU idling control, *machdep.idle* should be set to acpi if it is listed in *machdep.idle_available*.

dev.cpu.N.cx_supported

List of supported CPU idle states and their transition latency in microseconds. Each state has a type (e.g., C2). C1 is equivalent to the ia32 HLT instruction, C2 provides a deeper sleep with the same semantics, and C3 provides the deepest sleep but additionally requires bus mastering to be

disabled. States greater than C3 provide even more power savings with the same semantics as the C3 state. Deeper sleeps provide more power savings but increased transition latency when an interrupt occurs.

dev.cpu.N.cx_method
List of supported CPU idle states and their transition methods, as directed by the firmware.

hw.acpi.acline
AC line state (1 means online, 0 means on battery power).

hw.acpi.disable_on_reboot
Disable ACPI during the reboot process. Most systems reboot fine with ACPI still enabled, but some require exiting to legacy mode first. Default is 0, leave ACPI enabled.

hw.acpi.handle_reboot
Use the ACPI Reset Register capability to reboot the system. Some newer systems require use of this register, while some only work with legacy rebooting support.

hw.acpi.lid_switch_state
Suspend state (S1-S5) to enter when the lid switch (i.e., a notebook screen) is closed, or "NONE" (do nothing). Default is "NONE".

hw.acpi.power_button_state
Suspend state (S1-S5) to enter when the power button is pressed, or "NONE" (do nothing). Default is S5 (power-off nicely).

hw.acpi.reset_video
Reset the video adapter from real mode during the resume path. Some systems need this help, others have display problems if it is enabled. Default is 0 (disabled).

hw.acpi.s4bios
Indicate whether the system supports S4BIOS. This means that the BIOS can handle all the functions of suspending the system to disk. Otherwise, the OS is responsible for suspending to disk (S4OS). Most current systems do not support S4BIOS.

hw.acpi.sleep_button_state
Suspend state (S1-S5) to enter when the sleep button is pressed. This is usually a special function button on the keyboard. Default is S3 (suspend-to-RAM).

hw.acpi.sleep_delay

Wait this number of seconds between preparing the system to suspend and actually entering the suspend state. Default is 1 second.

hw.acpi.supported_sleep_state
Suspend states (S1-S5) supported by the BIOS.

S1 Quick suspend to RAM. The CPU enters a lower power state, but most peripherals are left running.

S2 Lower power state than S1, but with the same basic characteristics. Not supported by many systems.

S3 Suspend to RAM. Most devices are powered off, and the system stops running except for memory refresh.

S4 Suspend to disk. All devices are powered off, and the system stops running. When resuming, the system starts as if from a cold power on. Not yet supported by FreeBSD unless S4BIOS is available.

S5 System shuts down cleanly and powers off.

hw.acpi.verbose
Enable verbose printing from the various ACPI subsystems.

LOADER TUNABLES

Tunables can be set at the loader(8) prompt before booting the kernel or stored in */boot/loader.conf*. Many of these tunables also have a matching sysctl(8) entry for access after boot.

acpi_dsdt_load
Enables loading of a custom ACPI DSDT.

acpi_dsdt_name
Name of the DSDT table to load, if loading is enabled.

debug.acpi.disabled
Selectively disables portions of ACPI for debugging purposes.

debug.acpi.interpreter_slack
Enable less strict ACPI implementations. Default is 1, ignore common BIOS mistakes.

debug.acpi.max_threads

Specify the number of task threads that are started on boot. Limiting this to 1 may help work around various BIOSes that cannot handle parallel requests. The default value is 3.

debug.acpi.quirks

Override any automatic quirks completely.

debug.acpi.resume_beep

Beep the PC speaker on resume. This can help diagnose suspend/resume problems. Default is 0 (disabled).

hint.acpi.0.disabled

Set this to 1 to disable all of ACPI. If ACPI has been disabled on your system due to a blacklist entry for your BIOS, you can set this to 0 to re-enable ACPI for testing.

hw.acpi.ec.poll_timeout

Delay in milliseconds to wait for the EC to respond. Try increasing this number if you get the error "AE_NO_HARDWARE_RESPONSE".

hw.acpi.host_mem_start

Override the assumed memory starting address for PCI host bridges.

hw.acpi.install_interface, hw.acpi.remove_interface

Install or remove OS interface(s) to control return value of '_OSI' query method. When an OS interface is specified in *hw.acpi.install_interface*, _OSI query for the interface returns it is *supported*. Conversely, when an OS interface is specified in *hw.acpi.remove_interface*, _OSI query returns it is *not supported*. Multiple interfaces can be specified in a comma-separated list and any leading white spaces will be ignored. For example, "FreeBSD, Linux" is a valid list of two interfaces "FreeBSD" and "Linux".

hw.acpi.reset_video

Enables calling the VESA reset BIOS vector on the resume path. This can fix some graphics cards that have problems such as LCD white-out after resume. Default is 0 (disabled).

hw.acpi.serialize_methods

Allow override of whether methods execute in parallel or not. Enable this for serial behavior, which fixes "AE_ALREADY_EXISTS" errors for AML that really cannot handle parallel method execution. It is off by default since this breaks recursive methods and some IBMs use such code.

hw.acpi.verbose

Turn on verbose debugging information about what ACPI is doing.

hw.pci.link.%s.%d.irq

Override the interrupt to use for this link and index. This capability should be used carefully, and only if a device is not working with **acpi** enabled. "%s" is the name of the link (e.g., LNKA). "%d" is the resource index when the link supports multiple IRQs. Most PCI links only have one IRQ resource, so the below form should be used.

hw.pci.link.%s.irq

Override the interrupt to use. This capability should be used carefully, and only if a device is not working with **acpi** enabled. "%s" is the name of the link (e.g., LNKA).

DISABLING ACPI

Since ACPI support on different platforms varies greatly, there are many debugging and tuning options available.

For machines known not to work with **acpi** enabled, there is a BIOS blacklist. Currently, the blacklist only controls whether **acpi** should be disabled or not. In the future, it will have more granularity to control features (the infrastructure for that is already there).

To enable **acpi** (for debugging purposes, etc.) on machines that are on the blacklist, set the kernel environment variable *hint.acpi.0.disabled* to 0. Before trying this, consider updating your BIOS to a more recent version that may be compatible with ACPI.

To disable the **acpi** driver completely, set the kernel environment variable *hint.acpi.0.disabled* to 1.

Some i386 machines totally fail to operate with some or all of ACPI disabled. Other i386 machines fail with ACPI enabled. Disabling all or part of ACPI on non-i386 platforms (i.e., platforms where ACPI support is mandatory) may result in a non-functional system.

The **acpi** driver comprises a set of drivers, which may be selectively disabled in case of problems. To disable a sub-driver, list it in the kernel environment variable *debug.acpi.disabled*. Multiple entries can be listed, separated by a space.

ACPI sub-devices and features that can be disabled:

all Disable all ACPI features and devices.

acad (*device*) Supports AC adapter.

bus (*feature*) Probes and attaches subdevices. Disabling will avoid scanning the ACPI namespace entirely.

children (*feature*) Attaches standard ACPI sub-drivers and devices enumerated in the ACPI namespace. Disabling this has a similar effect to disabling "bus", except that the ACPI namespace will still be scanned.

button (*device*) Supports ACPI button devices (typically power and sleep buttons).

cmbat (*device*) Control-method batteries device.

cpu (*device*) Supports CPU power-saving and speed-setting functions.

ec (*device*) Supports the ACPI Embedded Controller interface, used to communicate with embedded platform controllers.

isa (*device*) Supports an ISA bus bridge defined in the ACPI namespace, typically as a child of a PCI bus.

lid (*device*) Supports an ACPI laptop lid switch, which typically puts a system to sleep.

mwait (*feature*) Do not ask firmware for available x86-vendor specific methods to enter Cx sleep states. Only query and use the generic I/O-based entrance method. The knob is provided to work around inconsistencies in the tables filled by firmware.

quirks (*feature*) Do not honor quirks. Quirks automatically disable ACPI functionality based on the XSDT table’s OEM vendor name and revision date.

pci (*device*) Supports Host to PCI bridges.

pci_link (*feature*) Performs PCI interrupt routing.

sysresource (*device*) Pseudo-devices containing resources which ACPI claims.

thermal (*device*) Supports system cooling and heat management.

timer (*device*) Implements a timecounter using the ACPI fixed-frequency timer.

video (*device*) Supports acpi_video(4) which may conflict with agp(4) device.

It is also possible to avoid portions of the ACPI namespace which may be causing problems, by listing the full path of the root of the region to be avoided in the kernel environment variable *debug.acpi.avoid*. The object and all of its children will be ignored during the bus/children scan of the namespace. The ACPI CA code will still know about the avoided region.

DEBUGGING OUTPUT

To enable debugging output, **acpi** must be compiled with **options ACPI_DEBUG**. Debugging output is separated between layers and levels, where a layer is a component of the ACPI subsystem, and a level is a particular kind of debugging output.

Both layers and levels are specified as a whitespace-separated list of tokens, with layers listed in *debug.acpi.layer* and levels in *debug.acpi.level*.

The first set of layers is for ACPI-CA components, and the second is for FreeBSD drivers. The ACPI-CA layer descriptions include the prefix for the files they refer to. The supported layers are:

ACPI_UTILITIES	Utility ("ut") functions
ACPI_HARDWARE	Hardware access ("hw")
ACPI_EVENTS	Event and GPE ("ev")
ACPI_TABLES	Table access ("tb")
ACPI_NAMESPACE	Namespace evaluation ("ns")
ACPI_PARSER	AML parser ("ps")
ACPI_DISPATCHER	Internal representation of interpreter state ("ds")
ACPI_EXECUTER	Execute AML methods ("ex")
ACPI_RESOURCES	Resource parsing ("rs")
ACPI_CA_DEBUGGER	Debugger implementation ("db", "dm")
ACPI_OS_SERVICES	Usermode support routines ("os")
ACPI_CA_DISASSEMBLER	Disassembler implementation (unused)
ACPI_ALL_COMPONENTS	All the above ACPI-CA components
ACPI_AC_ADAPTER	AC adapter driver
ACPI_BATTERY	Control-method battery driver
ACPI_BUS	ACPI, ISA, and PCI bus drivers
ACPI_BUTTON	Power and sleep button driver
ACPI_EC	Embedded controller driver
ACPI_FAN	Fan driver
ACPI_OEM	Platform-specific driver for hotkeys, LED, etc.
ACPI_POWER	Power resource driver
ACPI_PROCESSOR	CPU driver
ACPI_THERMAL	Thermal zone driver
ACPI_TIMER	Timer driver

ACPI_ALL_DRIVERS	All the above FreeBSD ACPI drivers

The supported levels are:

ACPI_LV_INIT	Initialization progress
ACPI_LV_DEBUG_OBJECT	Stores to objects
ACPI_LV_INFO	General information and progress
ACPI_LV_REPAIR	Repair a common problem with predefined methods
ACPI_LV_ALL_EXCEPTIONS	All the previous levels
ACPI_LV_PARSE	
ACPI_LV_DISPATCH	
ACPI_LV_EXEC	
ACPI_LV_NAMES	
ACPI_LV_OPREGION	
ACPI_LV_BFIELD	
ACPI_LV_TABLES	
ACPI_LV_VALUES	
ACPI_LV_OBJECTS	
ACPI_LV_RESOURCES	
ACPI_LV_USER_REQUESTS	
ACPI_LV_PACKAGE	
ACPI_LV_VERBOSITY1	All the previous levels
ACPI_LV_ALLOCATIONS	
ACPI_LV_FUNCTIONS	
ACPI_LV_OPTIMIZATIONS	
ACPI_LV_VERBOSITY2	All the previous levels
ACPI_LV_ALL	Synonym for "ACPI_LV_VERBOSITY2"
ACPI_LV_MUTEX	
ACPI_LV_THREADS	
ACPI_LV_IO	
ACPI_LV_INTERRUPTS	
ACPI_LV_VERBOSITY3	All the previous levels
ACPI_LV_AML_DISASSEMBLE	
ACPI_LV_VERBOSE_INFO	
ACPI_LV_FULL_TABLES	
ACPI_LV_EVENTS	
ACPI_LV_VERBOSE	All levels after "ACPI_LV_VERBOSITY3"
ACPI_LV_INIT_NAMES	
ACPI_LV_LOAD	

Selection of the appropriate layer and level values is important to avoid massive amounts of debugging output. For example, the following configuration is a good way to gather initial information. It enables debug output for both ACPI-CA and the **acpi** driver, printing basic information about errors, warnings, and progress.

```
debug.acpi.layer="ACPI_ALL_COMPONENTS ACPI_ALL_DRIVERS"
debug.acpi.level="ACPI_LV_ALL_EXCEPTIONS"
```

Debugging output by the ACPI CA subsystem is prefixed with the module name in lowercase, followed by a source line number. Output from the FreeBSD-local code follows the same format, but the module name is uppercased.

OVERRIDING YOUR BIOS BYTECODE

ACPI interprets bytecode named AML (ACPI Machine Language) provided by the BIOS vendor as a memory image at boot time. Sometimes, the AML code contains a bug that does not appear when parsed by the Microsoft implementation. FreeBSD provides a way to override it with your own AML code to work around or debug such problems. Note that all AML in your DSDT and any SSDT tables is overridden.

In order to load your AML code, you must edit */boot/loader.conf* and include the following lines.

```
acpi_dsdt_load="YES"
acpi_dsdt_name="/boot/acpi_dsdt.aml" # You may change this name.
```

In order to prepare your AML code, you will need the acpidump(8) and iasl(8) utilities and some ACPI knowledge.

COMPATIBILITY

ACPI is only found and supported on i386/ia32 and amd64.

SEE ALSO

kenv(1), acpi_thermal(4), device.hints(5), loader.conf(5), acpiconf(8), acpidump(8), config(8), iasl(8)

Compaq Computer Corporation, Intel Corporation, Microsoft Corporation, Phoenix Technologies Ltd., and Toshiba Corporation, *Advanced Configuration and Power Interface Specification*, http://acpi.info/spec.htm, August 25, 2003.

AUTHORS

The ACPI CA subsystem is developed and maintained by Intel Architecture Labs.

The following people made notable contributions to the ACPI subsystem in FreeBSD: Michael Smith, Takanori Watanabe <*takawata@jp.FreeBSD.org*>, Mitsuru IWASAKI <*iwasaki@jp.FreeBSD.org*>, Munehiro Matsuda, Nate Lawson, the ACPI-jp mailing list at <*acpi-jp@jp.FreeBSD.org*>, and many other contributors.

This manual page was written by Michael Smith <*msmith@FreeBSD.org*>.

BUGS

Many BIOS versions have serious bugs that may cause system instability, break suspend/resume, or prevent devices from operating properly due to IRQ routing problems. Upgrade your BIOS to the latest version available from the vendor before deciding it is a problem with **acpi**.