

# Linux ACPI Test Plan

---

## Revision 0.3

Last Save Date: 12/27/2005 3:03:00 PM

Last Changed By: Ling Yu

Key Contributors:     Ling Yu  
                           Len Brown; Shaohua Li; Luming Yu

## Revision History

Revision	Date	Author	Reason for Changes
0.1	12/14/05	Ling Yu	Initial Revision
0.3	12/27/05	Ling Yu	Added revision history section. Includes Len's some review comments. Convert the features list format.

## Overview

In order to provide the effective validation for the Linux ACPI drivers, this document defines the test environment, the features to be tested and test approaches. This test plan is tend to enhance when getting more inputs or requirements from community.

## Test Environment

ACPI patch Release:

<http://www.kernel.org/pub/linux/kernel/people/lenb/acpi/patches/release/>

Kernel: 2.6 and later

Platform: ia32, x86\_64, ia64

## Function Areas

The following function areas of ACPI driver will be covered by this test plan.

### 1 Power Management

- 1.1 S-state
- 1.2 C-state
- 1.3 P-state
- 1.4 T-state
- 1.5 Power and Battery
- 1.6 Thermal Zone

### 2 Configurations

- 2.1 ACPI event
- 2.2 AC adapter
- 2.3 Battery
- 2.4 ASUS(HotKey)

- 2.5 Generic Hotkey
- 2.6 Button
- 2.7 Fan
- 2.8 Video
- 2.9 Embedded controller
- 2.10 Hotpluggable devices

## Feature to be tested

The follow content defines and describes the feature details to be tested.

### 1 Power Management

#### 1.1 S-state

1.1.1 S-state file system interface;

1.1.2 Suspend to RAM, resume and check the keyboard, mouse, touchpad, video, USB, CDROM and network;

1.1.3 Suspend to Disk, resume and check the keyboard, mouse, touchpad, video, USB, CDROM and network. Need some pre-checking before sleep operation to isolate the hardware or other driver bugs from ACPI bug, such as USB, video and SATA;

1.1.4 For S3 and S4, suspend and use RTC wake to resume repeatedly, the iteration number should be more than 100.

#### 1.2 C-state

1.2.1 C-state file system interface

1.2.2 The every supported C-states can be entered on UP

1.2.3 The every supported C-states can be entered on SMP.

1.2.4 The C-states when AC adapter and battery are as power supplier

1.2.5 The C-states change when switching between AC and Battery

1.2.6 Performance of C-states on different platforms.

#### 1.3 P-state

1.3.1 P-state file system interface

1.3.2 Verify that the performance governor: set to performance governor and check the frequencies of all the CPUS. It should be the maximum supported freq.

1.3.3 Verify that the powersave governor Set to powersave governor and check the frequencies of all the CPUS. It should be the minimum supported freq.

1.3.4 Verify that the userspace governor Set to userspace governor and make sure, that it is set properly; set to all possible frequencies and check whether it is set properly.

1.3.5 Verify that the ondemand governor obeys the scaling max and min frequencies: set to ondemand governor and make sure it is set properly; Take all CPUs to idle and make sure the frequencies goes to the minimum; Keep all CPUs busy and make sure freq goes to the maximum; keep some CPUs busy and others idle and make sure the frequencies are set appropriately.

1.3.6 Verify the ondemand governor performance, including: run make -jn with and without DBS performance difference should not be more than 2%; run volanomark benchmark with and without DBS performance. A difference should not be more than 2%.

#### 1.4 T-state

1.4.1 T-state file system interface

1.4.2 Set the current throttling state for the CPU by writing the state number to `/proc/acpi/processor/CPU*/throttling`. Read it back to make sure the change works.

1.4.3 Measure increasing amount of the time under the situation that we give a certain amount of work and a T-state of the CPU.

1.4.4 When temperature is a concern, the T-state will help to keep your CPU running cooler.

1.4.5 T-state performance: observe any performance differences through the `/sys/devices/system/cpu/`

cpu\*/cpufreq/stats and /proc/acpi/processor/\*/power during controlled workloads.

## 1.5 Power and Battery

1.5.1 Measure the actual power on the machine through physically incrementing systems, such as using an Intel iPARK power meter on a dual Xeon, and measuring known workloads to verify that our performance/watt improves and does not get worse over time.

1.5.2 Measuring battery life on laptops using the battery life assessment tools.

## 1.6 Thermal Zone

1.6.1 Thermal Zone file system interface;

1.6.2 Monitor for each thermal zone by reading the file /proc/acpi/thermal\_zone/\*/temperature and adjust the system cooling mode by /proc/acpi/thermal\_zone/\*/cooling\_mode. Cooling mode can be critical, passive, or active. In each supported thermal zone and the active cooling mode, when a fan the temperature passes a critical point, the cooling device can be turned on (the fan speed can be increasing)

1.6.3 In the passive cooling mode, when the temperature is too hot, the devices can be put into a lower power state

1.6.4 In the Critical cooling, when the temperature passes one trip point, the system will take actions according to cooling policy.

1.6.5 Verify the TM1/TM2 thermal throttling functionality. TM1 takes half its clock cycles off, cooling the CPU and reducing power. TM2 throttling steps down processor clock speed and voltage via the ACPI

1.6.6 Make sure that the thermal sub-system works properly and never breaks.

## 2 Configuration

### 2.1 ACPI event

2.1.1 Operate the device to trigger some kinds of event to /proc/acpi/event, include buttons, battery, ac, and etc, the events can be exported to /proc/acpi/event and with correct format. (Devices include ac\_adapter, battery, button, container, embedded\_controller, fan, Hotkeys, lid, memory, pci\_bridge, pci\_irq\_routing, power, power\_resource, processor, sleep, system\_bus, system, thermal\_zone, video)

### 2.2 AC adapter

2.2.1 Driver load/unload

2.2.2 AC adapter information and state in ACPI interface /proc/acpi/ac\_adapter

2.2.3 AC adapter plug/play

### 2.3 Battery

2.3.1 Driver load/unload

2.3.2 Battery information and state in ACPI interface /proc/acpi/battery.

2.3.3 Battery plug/play

### 2.4 ASUS(HotKey)

2.4.1 Driver load/unload

2.4.2 File system interface.

2.4.3 Laptop Hotkey are functioned. (<http://sourceforge.net/projects/acpi4asus/>). This will be very system dependent.

### 2.5 Generic Hotkey

2.5.1 Driver load/unload

2.5.2 File system interface

2.5.3 Generic Hotkeys are functioned. This will be very system dependent.

### 2.6 Button

2.6.1 Driver load/unload

2.6.2 File system interface

2.6.3 Button control, including power button, sleep button and lid.

### 2.7 Fan

2.7.1 Driver load/unload

2.7.2 File system interface

2.7.3 Fan control through echo the D-state number to /proc/acpi/fan on the system which has ACPI fan control

## 2.8 Video

2.8.1 Driver load/unload

2.8.2 File system interface

2.8.3 Video control through echo state number to `/proc/video/VID/*/state` to change the state of the display.

## 2.9 Embedded controller

2.9.1 Driver load/unload

2.9.2 File system interface.

2.9.3 Embedded controller which manage or communicate with smart batteries and smart battery chargers.

## 2.10 Hotpluggable devices

2.10.1 ACPI Ejection Dependency Support. This feature has relationship with the mechanisms for handling dynamic insertion and removal of devices. `_EJD` evaluates the name of a device object on which the device that declared the object `_EJD` depends. Whenever the named device is ejected, the dependent device must also be ejected.

2.10.2 ACPI Module device hotplug. Module device is called "container" and typically used to describe CPUs, memory, root bridges, P2P bridges and etc.

## Test Approach

### 1. Automatic Basic Acceptance Testing (ABAT)

ABAT is expected to include a testing tool to dump/validate all the ACPI interface information under `/proc` and `/sys`, the testing of load/unload all ACPI drivers and some basic and automatic functional test suites, ABAT will give a summary about the ACPI driver basic health status.

### 2. Functional Test

Provide basic test suite for the ACPI features to let open source users to validate their ACPI driver on their system. These part of test suite might be manual and automatic.

### 3. Performance Test

Provide the performance measure on different platforms. Power management test should consider the performance test as much as possible.

### 4. Stress Test

Under enough quantities of task burdens and enough long working time, make sure that the ACPI driver works properly and never breaks.

### 5. Regression Test

Bug verification test to make sure no ACPI function regression. Regression test is a good way to make sure the ACPI driver development makes progress and never go backwards. Might need separate test cases to verify the specific bugs and need run them on the new releases to make sure the bugs won't appear any more.