ACPI Device Tree - Representation of ACPI Namespace

Copyright: © 2013, Intel Corporation
Author: Lv Zheng <lv.zheng@intel.com>
Credit: Thanks for the help from Zhang Rui <rui.zhang@intel.com> and Rafael J.Wysocki <rafael.j.wysocki@intel.com>.

Abstract

The Linux ACPI subsystem converts ACPI namespace objects into a Linux device tree under the /sys/devices/LNXSYSTEM:00 and updates it upon receiving ACPI hotplug notification events. For each device object in this hierarchy there is a corresponding symbolic link in the /sys/bus/acpi/devices.

This document illustrates the structure of the ACPI device tree.

ACPI Definition Blocks

The ACPI firmware sets up RSDP (Root System Description Pointer) in the system memory address space pointing to the XSDT (Extended System Description Table). The XSDT always points to the FADT (Fixed ACPI Description Table) using its first entry, the data within the FADT includes various fixed-length entries that describe fixed ACPI features of the hardware. The FADT contains a pointer to the DSDT (Differentiated System Descripition Table). The XSDT also contains entries pointing to possibly multiple SSDTs (Secondary System Description Table).

The DSDT and SSDT data is organized in data structures called definition blocks that contain definitions of various objects, including ACPI control methods, encoded in AML (ACPI Machine Language). The data block of the DSDT along with the contents of SSDTs represents a hierarchical data structure called the ACPI namespace whose topology reflects the structure of the underlying hardware platform.

The relationships between ACPI System Definition Tables described above are illustrated in the following diagram:

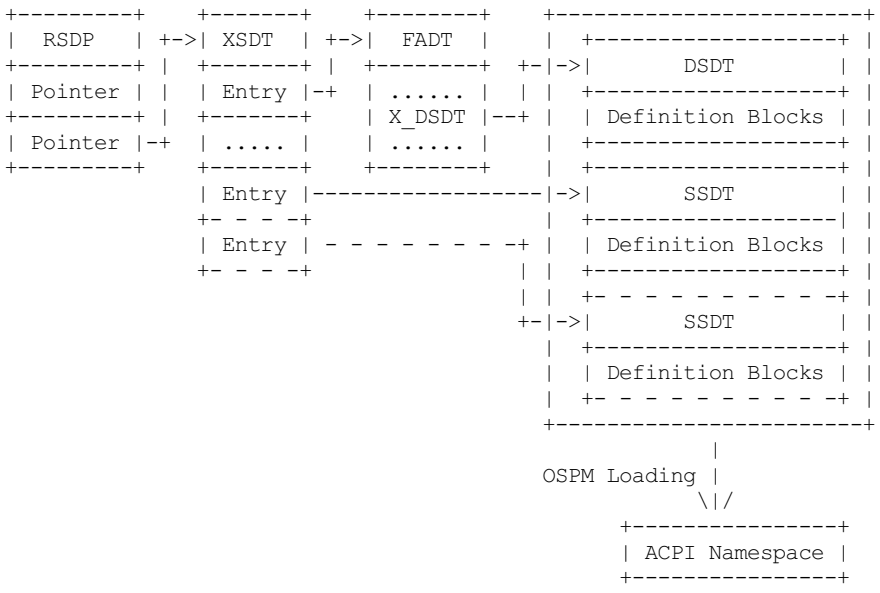


Figure 1. ACPI Definition Blocks

> **Note**
>
> RSDP can also contain a pointer to the RSDT (Root System Description Table). Platforms provide RSDT to enable compatibility with ACPI 1.0 operating systems. The OS is expected to use XSDT, if present.

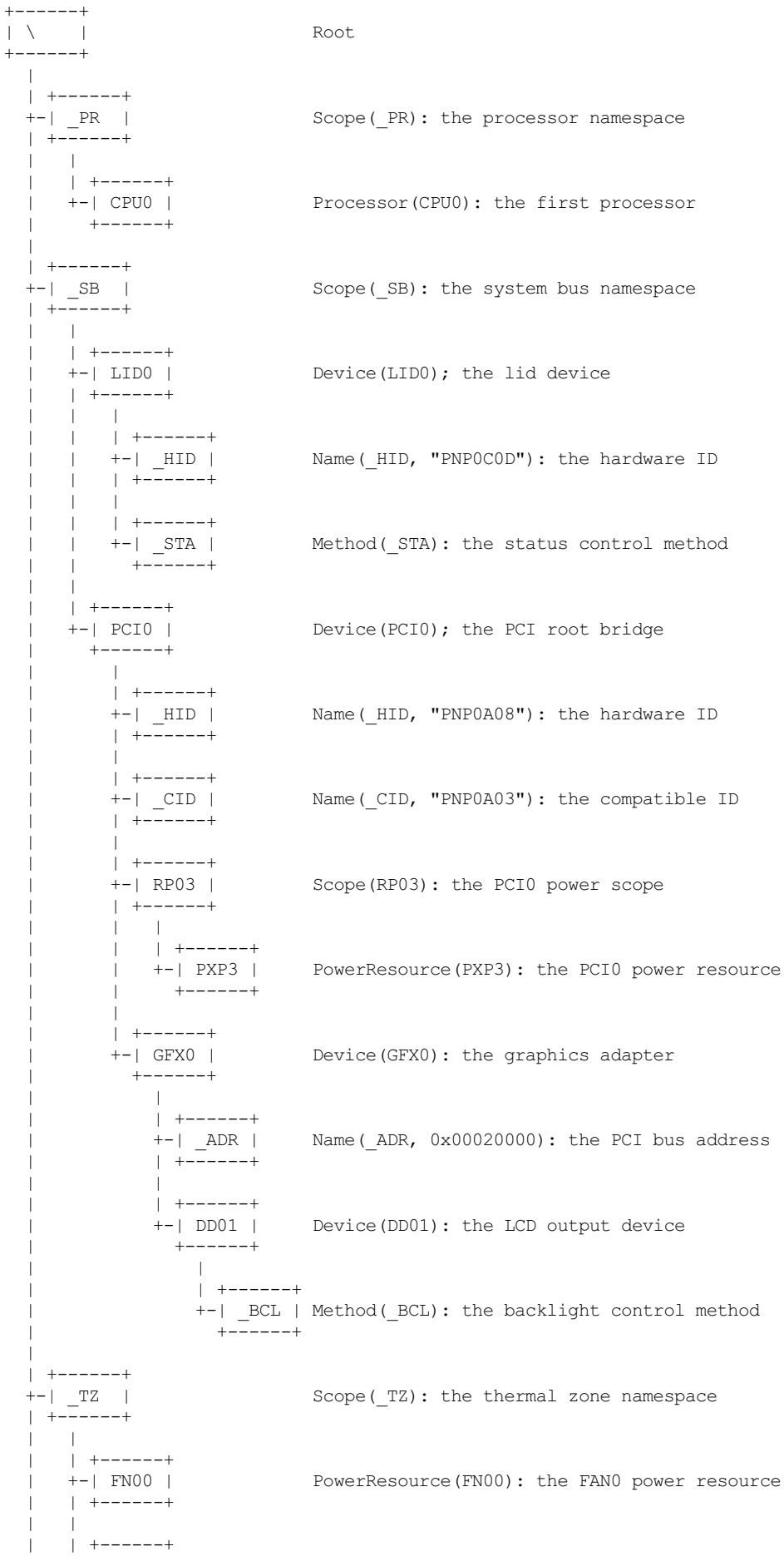
Example ACPI Namespace

All definition blocks are loaded into a single namespace. The namespace is a hierarchy of objects identified by names and paths. The following naming conventions apply to object names in the ACPI namespace:

1. All names are 32 bits long.
2. The first byte of a name must be one of 'A' - 'Z', '_'.
3. Each of the remaining bytes of a name must be one of 'A' - 'Z', '0' - '9', '_'.
4. Names starting with '_' are reserved by the ACPI specification.

5. The " symbol represents the root of the namespace (i.e. names prepended with " are relative to the namespace root).
6. The '^' symbol represents the parent of the current namespace node (i.e. names prepended with '^' are relative to the parent of the current namespace node).

The figure below shows an example ACPI namespace:



```
|   +-| FAN0 |             Device(FAN0): the FAN0 cooling device
|   | +------+
|   |   |
|   |   | +------+
|   |   +-| _HID |         Name(_HID, "PNP0A0B"): the hardware ID
|   |     +------+
|   |
|   | +------+
|   +-| TZ00 |             ThermalZone(TZ00); the FAN thermal zone
|     +------+
|
| +------+
+-| _GPE |                Scope(_GPE): the GPE namespace
  +------+
```

Figure 2. Example ACPI Namespace

Linux ACPI Device Objects

The Linux kernel's core ACPI subsystem creates struct acpi_device objects for ACPI namespace objects representing devices, power resources processors, thermal zones. Those objects are exported to user space via sysfs as directories in the subtree under /sys/devices/LNXSYSTM:00. The format of their names is <bus_id:instance>, where 'bus_id' refers to the ACPI namespace representation of the given object and 'instance' is used for distinguishing different object of the same 'bus_id' (it is two-digit decimal representation of an unsigned integer).

The value of 'bus_id' depends on the type of the object whose name it is part of as listed in the table below:

	Object/Feature	Table	bus_id
N	Root	xSDT	LNXSYSTM
N	Device	xSDT	_HID
N	Processor	xSDT	LNXCPU
N	ThermalZone	xSDT	LNXTHERM
N	PowerResource	xSDT	LNXPOWER
N	Other Devices	xSDT	device
F	PWR_BUTTON	FADT	LNXPWRBN
F	SLP_BUTTON	FADT	LNXSLPBN
M	Video Extension	xSDT	LNXVIDEO
M	ATA Controller	xSDT	LNXIOBAY
M	Docking Station	xSDT	LNXDOCK

Table 1. ACPI Namespace Objects Mapping

The following rules apply when creating struct acpi_device objects on the basis of the contents of ACPI System Description Tables (as indicated by the letter in the first column and the notation in the second column of the table above):

N:
: The object's source is an ACPI namespace node (as indicated by the named object's type in the second column). In that case the object's directory in sysfs will contain the 'path' attribute whose value is the full path to the node from the namespace root.

F:
: The struct acpi_device object is created for a fixed hardware feature (as indicated by the fixed feature flag's name in the second column), so its sysfs directory will not contain the 'path' attribute.

M:
: The struct acpi_device object is created for an ACPI namespace node with specific control methods (as indicated by the ACPI defined device's type in the second column). The 'path' attribute containing its namespace path will be present in its sysfs directory. For example, if the _BCL method is present for an ACPI namespace node, a struct acpi_device object with LNXVIDEO 'bus_id' will be created for it.

The third column of the above table indicates which ACPI System Description Tables contain information used for the creation of the struct acpi_device objects represented by the given row (xSDT means DSDT or SSDT).

The fourth column of the above table indicates the 'bus_id' generation rule of the struct acpi_device object:

_HID:
: _HID in the last column of the table means that the object's bus_id is derived from the _HID/_CID identification objects present under the corresponding ACPI namespace node. The object's sysfs directory will then contain the 'hid' and 'modalias' attributes that can be used to retrieve the _HID and _CIDs of that object.

LNXxxxxx:
: The 'modalias' attribute is also present for struct acpi_device objects having bus_id of the "LNXxxxxx" form (pseudo devices), in which cases it contains the bus_id string itself.

device:
: 'device' in the last column of the table indicates that the object's bus_id cannot be determined from _HID/_CID of the corresponding ACPI namespace node, although that object represents a device (for example, it may be a PCI device with _ADR defined and without _HID or _CID). In that case the string 'device' will be used as the object's bus_id.

Linux ACPI Physical Device Glue

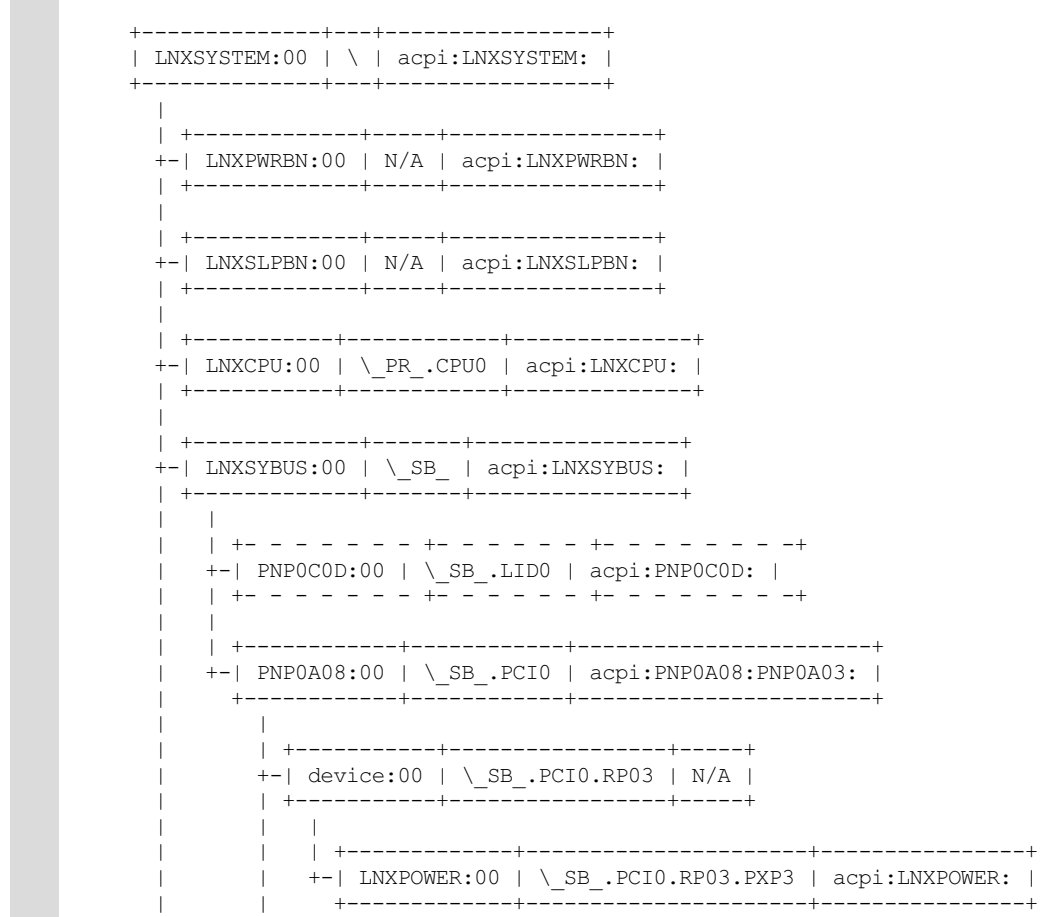
ACPI device (i.e. struct acpi_device) objects may be linked to other objects in the Linux' device hierarchy that represent "physical" devices (for example, devices on the PCI bus). If that happens, it means that the ACPI device object is a "companion" of a device otherwise represented in a different way and is used (1) to provide configuration information on that device which cannot be obtained by other means and (2) to do specific things to the device with the help of its ACPI control methods. One ACPI device object may be linked this way to multiple "physical" devices.

If an ACPI device object is linked to a "physical" device, its sysfs directory contains the "physical_node" symbolic link to the sysfs directory of the target device object. In turn, the target device's sysfs directory will then contain the "firmware_node" symbolic link to the sysfs directory of the companion ACPI device object. The linking mechanism relies on device identification provided by the ACPI namespace. For example, if there's an ACPI namespace object representing a PCI device (i.e. a device object under an ACPI namespace object representing a PCI bridge) whose _ADR returns 0x00020000 and the bus number of the parent PCI bridge is 0, the sysfs directory representing the struct acpi_device object created for that ACPI namespace object will contain the 'physical_node' symbolic link to the /sys/devices/pci0000:00/0000:00:02.0/ sysfs directory of the corresponding PCI device.

The linking mechanism is generally bus-specific. The core of its implementation is located in the drivers/acpi/glue.c file, but there are complementary parts depending on the bus types in question located elsewhere. For example, the PCI-specific part of it is located in drivers/pci/pci-acpi.c.

Example Linux ACPI Device Tree

The sysfs hierarchy of struct acpi_device objects corresponding to the example ACPI namespace illustrated in Figure 2 with the addition of fixed PWR_BUTTON/SLP_BUTTON devices is shown below:



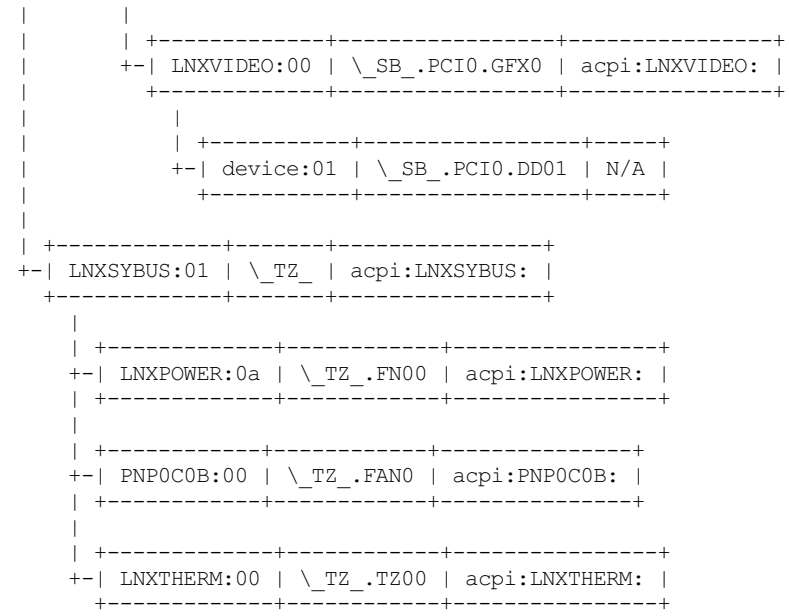


Figure 3. Example Linux ACPI Device Tree

Note

Each node is represented as "object/path/modalias", where:

1. 'object' is the name of the object's directory in sysfs.
2. 'path' is the ACPI namespace path of the corresponding ACPI namespace object, as returned by the object's 'path' sysfs attribute.
3. 'modalias' is the value of the object's 'modalias' sysfs attribute (as described earlier in this document).

Note

N/A indicates the device object does not have the 'path' or the 'modalias' attribute.