ACPI 6 and Linux*

Rafael J. Wysocki

Intel Open Source Technology Center

August 19, 2015
Outline

1. Introduction: Overview of ACPI
   - High Level Bits
   - Why ACPI?

2. Specification and Code Development
   - ACPI Specification Development Process
   - Reference Implementation
   - Linux/ACPI

3. ACPI 6 Coverage in Linux
   - Most Significant ACPI 6 Material
   - Other Changes in ACPI 6

4. Resources
ACPI: Advanced Configuration and Power Interface

Platform description + high-level interface for driving platform hardware.
What Is Covered By ACPI

- Enumeration of platform hardware.
- Signaling/handling of platform-specific events.
- Carrying out platform-specific operations.
ACPI Complexity
Who Needs/Wants (Things Like) ACPI

Whoever ships binary OS images to customers.

OS images that are platform-independent for real
- No need to include platform-specific code for every platform.
- No need to ship a separate (binary) kernel for every platform.

Platform firmware validation becomes a problem
Traditionally addressed by certification programs.
ACPI Is Taken Care Of By The UEFI Forum

ASWG: ACPI Specification Working Group
- Team responsible for the ACPI specification within the UEFI Forum.
- Drives ACPI specification development process.
- Produced ACPI specification revisions 5.1 and 6.0.

ECR: Engineering Change Request
- Formal specification change proposal.
- May be submitted by the UEFI Forum members.
- Goes under discussion in the ASWG.
ACPI Specification Development Work Flow

ACPI Specification Working Group

Final Specification Draft

Feedback

UEFI Member
ECR

Feedback

UEFI Forum Board

Final Draft For Review

Feedback

All UEFI Members / Legal

Updated Final Draft

Release
As Open As It Can Be

Legal concern (theoretical threat)
Possible “injection” of patent-contaminated material into the specification.

Remedy
New specification material cannot be shared with entities that are not UEFI Forum members until approved by the board.

What is possible
- Linux developers can contribute to the ACPI specification.
- Linux developers can influence ACPI material before it is adopted.
- Clarifications may be requested.
The ACPI CA Project

ACPI Component Architecture (https://acpica.org)

OS-independent reference implementation of the ACPI specification.

1. Common in-kernel code (shared by multiple OSes including Linux).
   - ACPI namespace representation
   - AML execution (AML interpreter)
   - Parsing of ACPI tables
   - Handling of events

2. AML compiler and disassembler.

3. ACPI-related utilities.

Linux kernel’s ACPICA code in sync with the upstream (Intel OTC).
ACPI A Is Open Source

Open Source project

- Source code on GitHub (https://github.com/acpica/acpica)
- Maintainers: Bob Moore, Lü Zheng, David Box
- Primary license: BSD (https://acpica.org/Licensing)

ACPI and Linux

- Semi-automated generation of ACPIA Linux patches.
- ACPIA code in Linux under the dual BSD/GPLv2 license.
- Linux patches are generally not applicable to upstream ACPIA.
Linux ACPI Code Beyond ACPICA

ACPI core
- Linux-specific support code used by ACPICA.
- ACPI-based device enumeration.
- ACPI part of the generic device properties API.
- Other Linux-specific generic ACPI code (PM, hotplug, thermal etc).
- Maintainer: Rafael Wysocki

ACPI drivers
Protocols defined by the specification: battery, fan, thermal, button, dock
Device Drivers Using ACPI

ACPI support in subsystems

- Enumeration of CPUs, interrupt controllers etc. (arch code)
- PCI
- USB
- SATA
- I2C
- GPIO

x86 platform drivers (maintained by Darren Hart)

Vendor-specific protocols based on ACPI.
Linux ACPI Support Structure

- ACPI Specification
- ACPICA
- Architecture-Specific Code
- ACPI Core
- Device Drivers
- ACPI 6 and Linux
Most Significant Changes Overview

Two types of changes
- Support for new platforms (new hardware types, more complexity).
- Improvements for the existing users.

ACPI 6 big-ticket items
- Support for C-style expressions in ASL (compiler).
- Non-Volatile Memory Firmware Interfaces.
- Low Power Idle (LPI).
- Device power management update.
Non-Volatile Memory Firmware Interfaces

NFIT: NVDIMM Firmware Interface Table
Non-volatile memory modules information and status at boot time.

NVDIMM device objects

1. NVDIMM interface device (ACPI0012) in the _SB scope.
   - _FIT (non-volatile memory modules information at run time).
   - _DSM (NVDIMM implementation specific).
   - NFIT Update Notifications.

2. NVDIMM devices (NFIT Device Handle returned by _ADR).
   - Under the NVDIMM interface device.
   - _DSM (NVDIMM implementation specific).
   - NFIT Health Event Notifications.
Non-Volatile Memory Modules Hotplug Support

Additional memory module object (PNP0C80) required for each NVDIMM (for Device Check notifications, _STA, and _CRS).
ACPI NFIT Protocol And Linux

Auxiliary documents (http://pmem.io/documents)

The _DSM methods require additional specification.

Linux support (v4.2)

- ACPICA: All what’s needed.
- Linux/ACPI: New ACPI driver for NVDIMMs (Dan Williams).
Low Power Idle (LPI)

Hierarchical representation of idle states
Idle states of CPU cores, CPU clusters (packages), groups of CPU clusters.

Key observation
Idle states at different levels of the hierarchy are not independent.
Processor Containers And Lists Of Idle States

Processor Container Device (ACPI0010)

- May contain other Processor Containers or Processors (ACPI0007).
- _LPI (Low Power Idle States).
- _RDI (Resource Dependencies for Idle) at the whole system level.
- _STA (Status).

The Low Power Idle States object

- _LPI: List of available idle states (in power consumption order).
- May be present under Processor Container or Processor Devices.
Low Power Idle Example

![Diagram of ACPI 6 coverage in Linux showing low power idle states for different components like CPU0, CPU1, CPU2, CPU3, CLU0, CLU1, and PROC. Each component has layers indicating different ACPI states (State 1, State 2, State 3).]
Platform Coordinated And OS Initiated LPI

Platform Coordinated LPI
- Platform responsible for the coordination of idle states.
- States requested for all levels of the hierarchy from each Processor.

OS Initiated LPI
- Last underlying Processor going idle triggers state selection.
- Requires software tracking of Processor states.

Linux support (v4.2)
- ACPICA: All what’s needed.
- Linux/ACPI: Not supported (work in progress by Linaro).
Device PM Update in ACPI 6

Clarification of the $D3_{hot}/D3_{cold}$ meaning

- $D3_{cold}$ only available if $_PR3$ is present.
- $D3_{hot}$ always available (if PM is supported at all).

Power state change sequence specification update

Power up : Turn on power resources and evaluate $_PS0$ (if present).
Power down : Evaluate $_PSx$ (if present) and turn off power resources.

Linux support (v4.2)

Linux/ACPI: Updated to follow ACPI 6 (Rafael Wysocki).
ACPI 6: Other Significant Changes

- USB-C Connection support in _UPC.
- New Thermal Zone objects (_MTL, _TSN).
- Standby Thermal Trip Point (_CR3).
- New Battery object (_BTH) and _BIX extension (swapping flag).
- Platform-specific device reset support (_RST, _PRR).
- Generic Buttons Device (ACPI0011) and a related _DSD UUID.
- Disambiguation of _REV.
- ASL helper macro for _PLD.
- CPPC and PCC updates.
- Printf and Fprintf debug macros (ASL).
- Improved disassembly of Control Method invocations (AML).
Linux Support Status

- Upstream ACPICA covers new ACPI 6 material entirely.
  - That ACPICA code will appear in Linux v4.2.
- Linux/ACPI will cover new ACPI 6 material when necessary.
Conclusion

- ACPI 6: The first major ACPI revision from the UEFI Forum.
- New specification development process (more open than ever).
- New interesting major features (NFIT, LPI).
- Full ACPI 6 support in the upstream ACPICA.
- Support in Linux/ACPI being added as needed.
Thanks!

Questions?
References

Documentation And Source Code

- https://acpica.org
- http://pmem.io/documents
- Documentation/acpi/
- Documentation/nvdimm/
- include/acpi/
- include/linux/acpi.h
- drivers/acpi/
- drivers/nvdimm/
- drivers/pci/pci-acpi.c
- drivers/gpio/gpiolib-acpi.c
- drivers/ata/libata-acpi.c
- drivers/usb/core/usb-acpi.c
- drivers/i2c/i2c-core.c
- arch/x86/kernel/acpi/
- arch/ia64/kernel/acpi.c
- arch/arm64/kernel/acpi.c