


[UEFI] Learn UEFI by RPI4 -- 2. ACPI

 UEFI This column includes this content

2 articles

Subscribe to our column

Continuing from the last article: [UEFI] Learn UEFI by RPI4 -- 1. Prepare and Build
In fact, ACPI is no longer basic content for UEFI. The reason I put it here is mainly because I have been reading about ACPI recently.

Introduction to ACPI

ACPI (Advanced Configuration and Power Interface) is a power management standard used to control the configuration and power management of computer hardware. ACPI was originally developed by Intel, Microsoft, Toshiba and other companies to replace old power management standards such as APM (Advanced Power Management) and PnP (Plug and Play). ACPI enables more precise and advanced power management functions by introducing a unified interface between the operating system and hardware. It includes a set of specifications, a set of firmware, and a set of operating system drivers that can coordinate the use and configuration of hardware resources such as processors, power supplies, memory, and peripherals. The ACPI standard defines a language for describing computer hardware and software configuration, called ACPI Description Language (ASL). With ASL, system designers can define information such as hardware configuration, device connections, and power management policies. The operating system reads this information and adjusts the behavior of the hardware accordingly.

The main functions of ACPI include:

- Power Management: ACPI can control and manage the power usage of the computer, including energy saving mode, standby mode, and hibernation mode. It can automatically adjust the power settings according to the user's needs and the status of the system to extend battery life and reduce energy consumption.
- Device configuration and connection: ACPI can detect and configure the hardware devices in the computer, including the processor, memory, peripherals, etc. It can also implement hot-plugging, that is, plugging or unplugging devices at runtime without restarting the computer.
- System status management: ACPI can track and manage the status of the system, including power on, power off, and restart. It can also monitor hardware failures and errors, and provide corresponding error handling and recovery mechanisms.

Several tables that must exist in ACPI

In the ACPI (Advanced Configuration and Power Interface) specification, the following tables must be included:

- RSDT (Root System Description Table) or XSDT (Extended System Description Table): These tables contain pointers to all other ACPI tables in the system. **XSDT is an extension of RSDT and supports 64-bit addresses, while RSDT only supports 32-bit addresses.**
- DSDT (Differentiated System Description Table): This is one of the most important ACPI tables, which contains a large number of AML (ACPI Machine Language) codes used to describe the configuration and power management functions of the system hardware. The DSDT table allows the operating system to query and control the power status of hardware devices.
- FADT (Fixed ACPI Description Table): Provides pointers to other important ACPI tables and contains system-level power management and configuration information, such as system power state, sleep button functionality, wake-up events, etc.
- MADT (Multiple APIC Description Table): describes the interrupt controller information in the system, including the configuration of local APIC and I/O APIC. This is necessary for the operating system to handle hardware interrupts correctly.
- SSDT (Secondary System Description Table): SSDT is usually used to supplement the information in the DSDT table and provide additional device descriptions. A system can have multiple SSDT tables.

1. Root System Description Pointer (RSDP)

The Root System Description Pointer (**RSDP**) structure is located in the system's memory address space and is set by the platform firmware. This structure contains the address of the Extended System Description Table (**XSDT**), which references other description tables that provide data to OSPM, giving it knowledge about the underlying system implementation and configuration (see Root System Description Pointers and Tables). All system description tables begin with the same header. The RSDT (Root System Description Table) is a key data structure in ACPI that contains pointers to all other system description tables. The main role of the RSDT is to act as a directory to help the operating system find and parse other ACPI tables.

Structure : The structure of the RSDT table consists of a standard ACPI table header and an array of pointers, each of which points to another ACPI table.

Function : The RSDT is used to locate and access other ACPI tables, such as FADT, DSDT, MADT, and SSDT.

Verification : The RSDT table needs to be verified by checksums to ensure its integrity and correctness [2][6].

- The Extended System Description Table (XSDT) points to other tables in memory. It is always the first table and points to the Fixed ACPI Description Table (**FADT**). The data in this table includes various fixed-length entries that describe the fixed ACPI capabilities of the hardware.
- The FADT table always references the Differentiated System Description Table (DSDT), which contains information and descriptions of various system functions. The relationship between these tables is shown in the Description Table Structure.

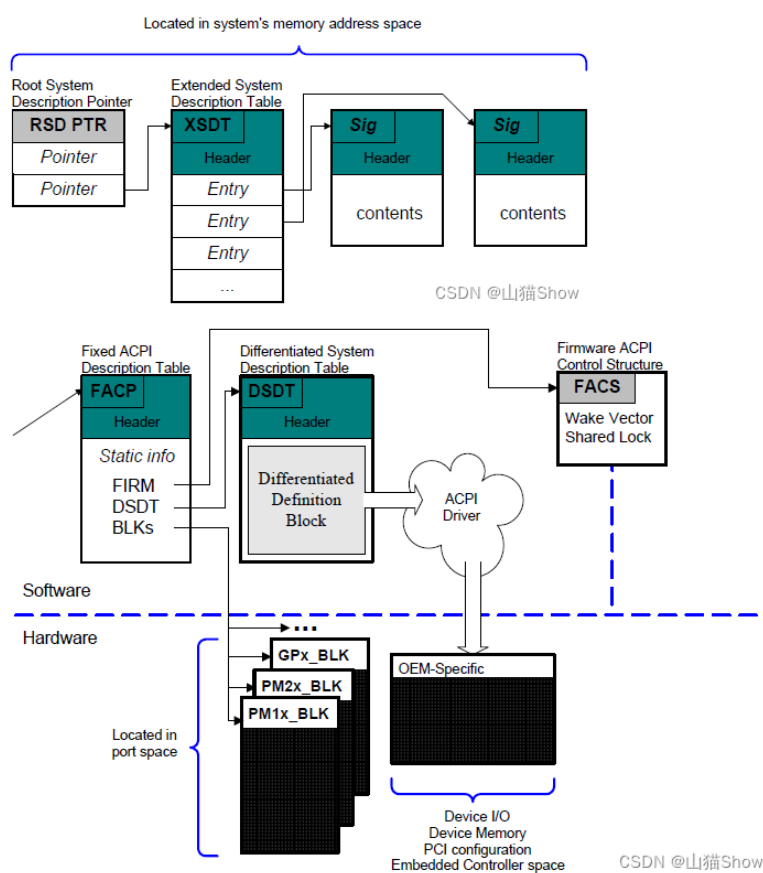
c

AI generated projects 登录复制 run

```
1 struct ACPISTHeader {
2     char Signature[4]; // 表签名
3     uint32_t Length; // 表长度
4     uint8_t Revision; // 版本
5     uint8_t Checksum; // 校验和
6     char OEMID[6]; // OEM ID
```

```
7 | char OEMTableID[8]; // OEM 表 ID
8 | uint32_t OEMRevision; // OEM 版本
9 | uint32_t CreatorID; // 创建者 ID
10 | uint32_t CreatorRevision; // 创建者版本
11 |};
```

收起 ^



2. DSDT (Differentiated System Description Table)

DSDT (Differentiated System Description Table) is a main table in ACPI that is used to describe the system's peripheral devices. DSDT contains detailed information about the device, such as I/O ports, IRQ, memory mapping, etc.

- **Structure** : The DSDT table consists of a standard ACPI header and an AML (ACPI Machine Language) code.
- **Function** : DSDT is used to define the hardware devices in the system and their power management functions. For example, when the operating system needs to shut down the system, it will look for the `_S5` object in DSDT to perform the shutdown operation.
- **Generation** : Hardware manufacturers use the ASL (ACPI Source Language) compiler to generate the AML bytecode for DSDT[5][6].

asl

AI generated projects

登录复制

```
1 | DefinitionBlock ("", "DSDT", 2, "OEMID", "OEMTABLE", 0x00000001)
2 | {
3 |     Device (DEV0)
4 |     {
5 |         Name (_HID, EisaId ("PNP0C0F")) // 设备ID
6 |         Name (_UID, 0x01) // 唯一ID
7 |         Method (_STA, 0, NotSerialized) // 状态方法
8 |         {
9 |             Return (0x0F) // 设备存在且已启用
10 |        }
11 |        Method (_PS0, 0, NotSerialized) // 电源状态0 (D0)
12 |        {
13 |            // 启动设备的代码
14 |        }
15 |        Method (_PS3, 0, NotSerialized) // 电源状态3 (D3)
16 |        {
17 |            // 关闭设备的代码
18 |        }
19 |    }
20 | }
```

收起 ^

3. FADT (Fixed ACPI Description Table)

FADT (Fixed ACPI Descriptor Table) is a data structure in ACPI that contains information about a fixed block of registers related to power management.

- **Structure** : The FADT table includes a standard ACPI header and multiple fields that provide various hardware information required for power management.

- **Function** : FADT is used to define the fixed hardware characteristics of the system, such as power management registers, system control interrupts (SCI), etc. It also contains a pointer to the DSDT.
- **Fields** : The fields of the FADT table include FirmwareCtrl, Dsdt, PreferredPowerManagementProfile, etc. [4][6].

cAI generated projects登录复制run

```
1 struct FADT {
2     struct ACPISDTHeader h;
3     uint32_t FirmwareCtrl;
4     uint32_t Dsdt;
5     uint8_t Reserved;
6     uint8_t PreferredPowerManagementProfile;
7     uint16_t SCI_Interrupt;
8     uint32_t SMI_CommandPort;
9     uint8_t AcpiEnable;
10    uint8_t AcpiDisable;
11    uint8_t S4BIOS_REQ;
12    uint8_t PSTATE_Control;
13    uint32_t PM1aEventBlock;
14    uint32_t PM1bEventBlock;
15    uint32_t PM1aControlBlock;
16    uint32_t PM1bControlBlock;
17    uint32_t PM2ControlBlock;
18    uint32_t PMTimerBlock;
19    uint32_t GPE0Block;
20    uint32_t GPE1Block;
21    uint8_t PM1EventLength;
22    uint8_t PM1ControlLength;
23    uint8_t PM2ControlLength;
24    uint8_t PMTimerLength;
25    uint8_t GPE0Length;
26    uint8_t GPE1Length;
27    uint8_t GPE1Base;
28    uint8_t CStateControl;
29 };
```

收起 ^

4. MADT (Multiple APIC Description Table)

MADT (Multiple APIC Description Table) is a table in ACPI that provides information about the system interrupt controller and is mainly used in multi-processor systems.

- **Structure** : The MADT table consists of a standard ACPI header and a series of variable-length records that describe the interrupt devices in the system.
- **Function** : MADT is used to enumerate the processors and interrupt controllers in the system to help the operating system configure and manage interrupts.
- **Record type** : The record types in the MADT table include processor local APIC, I/O APIC, interrupt source override, etc. [3][6].

cAI generated projects登录复制run

```
1 struct MADT {
2     struct ACPISDTHeader h;
3     uint32_t LocalAPICAddress;
4     uint32_t Flags;
5     // 可变长度的记录
6 };
```

5. SSDT (Secondary System Description Table)

SSDT (Secondary System Description Table) is a table in ACPI that supplements the DSDT, providing additional AML codes to interact with devices.

- **Structure** : The structure of the SSDT table is similar to that of the DSDT, consisting of a standard ACPI header and a section of AML code.
- **Function** : SSDT is used to define system components and functions not included in DSDT, and to provide additional device definition and power management functions.
- **Usage** : SSDT is often used to define hot-pluggable devices, dynamically loaded devices, etc. [1][6].

aslAI generated projects登录复制

```
1 DefinitionBlock ("", "SSDT", 2, "OEMID", "OEMTABLE", 0x00000001)
2 {
3     Device (DEV1)
4     {
5         Name (_HID, EisaId ("PNP0C0F")) // 设备ID
6         Name (_UID, 0x02) // 唯一ID
7         Method (_STA, 0, NotSerialized) // 状态方法
8         {
9             Return (0x0F) // 设备存在且已启用
10        }
11        Method (_PS0, 0, NotSerialized) // 电源状态0 (D0)
12        {
13            // 启动设备的代码
14        }
15        Method (_PS3, 0, NotSerialized) // 电源状态3 (D3)
16        {
17            // 关闭设备的代码
18        }
19    }
```

History of ACPI

The first version of the ACPI standard was released in December 1996, supporting 16, 24, and 32-bit addressing spaces. In August 2000, ACPI version 2.0 introduced 64-bit address support and support for multi-processor workstations and servers. Since then, the ACPI standard has continued to evolve, adding support for SATA interfaces, PCI Express buses, USB 3.0, ARM architectures, and more. The latest ACPI 6.5 version was released in August 2022.

ACPI Architecture

The ACPI architecture consists of several key components, including ACPI tables, ACPI BIOS, and ACPI registers.

ACPI Table

ACPI uses several tables to store information about hardware configuration and system status. These tables are critical for the operating system to understand hardware capabilities and manage the hardware. The main ACPI tables include:

- **DSDT (Differentiated System Description Table)** : Contains most of the ACPI data for the system, including definitions of most hardware components and their power management functions.
- **SSDT (Secondary System Description Table)** : Provides additional definitions of system components not included in DSDT.
- **FADT (Fixed ACPI Description Table)** : Provides static information required for the correct operation of various hardware components, including system-level information and pointers to other tables.
- **MADT (Multiple APIC Description Table)** : Contains information about the system interrupt controller and is mainly used in multi-processor systems.

ACPI BIOS

The system firmware or BIOS contains the ACPI implementation, providing the initial ACPI tables and interfaces required by the operating system to take over management of system resources. The ACPI BIOS is responsible for bootstrapping the ACPI environment before the operating system takes over.

ACPI Registers

ACPI hardware registers are divided into fixed hardware registers and general hardware registers. Fixed hardware registers need to implement the interface defined by ACPI, while general hardware registers are used for any events generated by value-added hardware. The ACPI register model includes status/enable registers and control registers.

Functions of ACPI

ACPI provides a number of key features that facilitate device management, power efficiency, and system responsiveness in modern computing systems. These features include:

- **Power management** : ACPI defines how to manage device power states (D states), processor states (C states and P states), and system states (S states, such as sleep and hibernation).
- **Event Handling** : ACPI defines mechanisms for handling various system events related to power, thermal management, and other system functions.
- **Thermal Management** : ACPI allows the operating system to dynamically adjust system policies based on temperature changes to ensure optimal performance and power usage.

ACPI Architecture Diagram

To better understand the architecture of ACPI (Advanced Configuration and Power Interface), we can use an architecture diagram to show its main components and their interrelationships. The following is a brief description and diagram of the ACPI architecture.

Main components of the ACPI architecture

1. ACPI Tables :

- **DSDT (Differentiated System Description Table)** : Contains most of the ACPI data for the system, including definitions of most hardware components and their power management functions.
- **SSDT (Secondary System Description Table)** : Provides additional definitions of system components not included in DSDT.
- **FADT (Fixed ACPI Description Table)** : Provides static information required for the correct operation of various hardware components, including system-level information and pointers to other tables.
- **MADT (Multiple APIC Description Table)** : Contains information about the system interrupt controller and is mainly used in multi-processor systems [2][3][4][5].

2. ACPI BIOS :

- The system firmware or BIOS contains the ACPI implementation, providing the initial ACPI tables and interfaces required by the operating system to take over system resource management. The ACPI BIOS is responsible for bootstrapping the ACPI environment before the operating system takes over [4].

3. ACPI Registers :

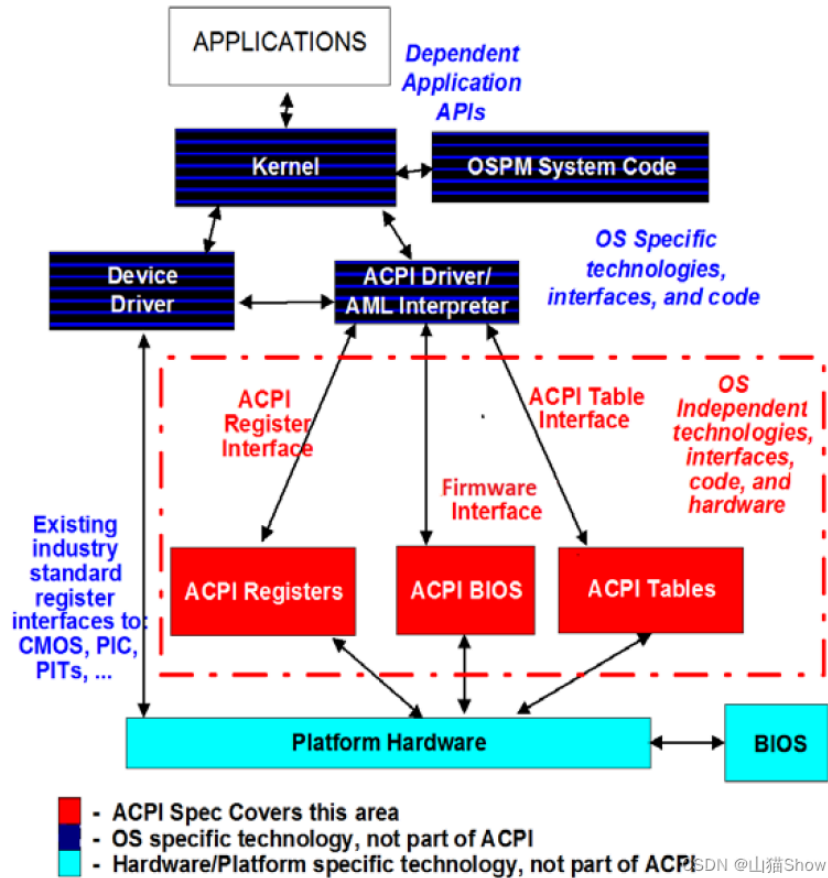
- ACPI hardware registers are divided into fixed hardware registers and general hardware registers. Fixed hardware registers need to implement the interface defined by ACPI, while general hardware registers are used for any events generated by value-added hardware [4].

4. AML (ACPI Machine Language) :

- ACPI tables are encoded in a bytecode language called ACPI Machine Language (AML). The ACPI component in the operating system interprets the AML bytecode, allowing the operating system to directly manage system hardware resources[3][4].

ACPI architecture diagram

The following is a diagram of the ACPI architecture, showing its main components and how they relate to each other:



Detailed description

- **ACPI tables** : These tables store information about hardware configuration and system status, and the operating system uses these tables to understand hardware capabilities and manage them.
- **ACPI BIOS** : Loads the ACPI table when the system starts and provides the initial ACPI environment.
- **ACPI registers** : used to manage hardware events and status.
- **AML** : AML codes in the ACPI tables are interpreted by the operating system to define events, device configuration, and power states.

ACPI Code Example

The following is a simple ACPI code example that shows how to define the power management capabilities of a device using ACPI Machine Language (AML):

```
asl
1 DefinitionBlock ("", "DSDT", 2, "OEMID", "OEMTABLE", 0x00000001)
2 {
3     Device (DEV0)
4     {
5         Name (_HID, EisaId ("PNP0C0F")) // Device ID
6         Name (_UID, 0x01) // Unique ID
7         Method (_STA, 0, NotSerialized) // Status method
8         {
9             Return (0x0F) // Device is present and enabled
10        }
11        Method (_PS0, 0, NotSerialized) // Power state 0 (D0)
12        {
13            // Code to power on the device
14        }
15        Method (_PS3, 0, NotSerialized) // Power state 3 (D3)
16        {
17            // Code to power off the device
18        }
19    }
20 }
```

Summarize

ACPI is a key system-level interface and power management specification that enables efficient power management and configuration by unifying and standardizing the interaction between the operating system and hardware. It not only covers power management, but also includes functions such as system event handling, thermal management, and device plug-and-play. Through ACPI, the operating system can more finely control system resources, improve system flexibility and power efficiency.

