PHYTIUM 飞腾

飞腾系统
ACPI 描述规范

(V1.2)

2020 年 12 月

天津飞腾信息技术有限公司

www.phytium.com.cn

当前版本

文件标识	**P-U-SW-ACPI**
当前版本	**1.2**
完成日期	2020.12.25

版本历史

版本	修订时间	修订人	修订章节	修订内容
1.0	2019.10.09			第一个正式发布的版本
1.1	2020.05.21		5.5；附录 A；附录 B；附录 C；附录 D	增加 SCPI、CAN、OP-TEE、SPI、SCMI、PMU、Thermal Zone、IPMI；修改 clock；修改 HID 定义；增加对设备状态动态更新的支持；部分设备增加_CCA 属性描述；GMAC 增加 mdc_clock_selection
1.2	2020.12.25			修正 fixed-clock 以及 I2C 的 HID;增加 GPIO 的 package 描述;修改 gmac phy-mode 的参考值;增加 can 的 clock-frequency 描述；增加 Trusted OS 的 smc method;增加 PPTT 表描述；FADT 表增加对台式机、笔记本、服务器的描述

目录

1 范畴.....1

2 定义与缩写.....2

2.1 定义.....2

2.2 缩写.....2

3 参考文献.....3

4 飞腾系统 ACPI 表.....4

4.1 DSDT.....4

4.2 FADT.....4

4.3 GTDT.....5

4.4 MADT.....5

4.5 MCFG.....6

4.6 RSDP.....7

4.7 SPCR.....7

4.8 XSDT.....7

4.9 IORT.....8

4.10 PPTT.....8

5 飞腾系统设备 ACPI 描述.....11

5.1 概述.....11

5.2 电源域划分.....11

5.3 处理器低功耗状态.....11

5.4 PCIE 控制器.....12

5.5 处理器 SOC 设备.....12

附录 A SOC 设备的 ACPI 描述.....15

A.1 UART.....15

A.2 Fixed-clock.....16

A.3 GPIO.....17

A.4 I2C.....18

A.5 Watchdog 20
A.6 GMAC 20
A.7 SDC 22
A.8 HDAudio 23
A.9 LPC 23
A.10 SCPI 24
A.11 CAN 26
A.12 Trusted OS 28
A.13 SPI 28
A.14 SCMI 30
A.15 PMU 31
A.16 Temperature Sensor 32
A.17 IPMI 33
A.17.1 Name Space 34
A.17.2 SPMI 34
附录 B CPU 的 ACPI 描述 37
附录 C PCI Express 的 ACPI 描述 40
附录 D EC 的 ACPI 描述 46

表目录

表 4-1 Preferred PM Profile 字段取值及含义 5
表 4-2 PPTT Table 8
表 5-1 Phytium Device HID 列表 13
表 附 A-0-1 Watchdog Timer Flags 20
表 附 A-0-2 SPMI Table 35

1 范畴

此文档适用于基于飞腾处理器的系统平台，为支持 ACPI 的固件设计提供参考。

2 定义与缩写

2.1 定义

2.2 缩写

ACPI	Advanced Configuration and Power Interface

3 参考文献

[1] UEFI, Advanced Configuration and Power Interface Specification, Version 6.3.

[2] ARM, ARM Functional Fixed Hardware Specification-ARM DEN 0048A

[3] Phytium, Phytium Base Firmware 接口规范

[4] PCI-SIG, PCI Firmware Specification

[5] ARM, IO Remapping Table Platform Design Document

[6] Phytium, 飞腾平台 Embedded Controller 接口规范

[7] Phytium, FT-2000/4 软件编程手册

[8] UEFI, http://uefi.org/acpi_id_list

4 飞腾系统 ACPI 表

飞腾处理器系统平台采用精简硬件 ACPI 模式（Hardware-reduced ACPI mode），在该模式中，没有使用硬件实现 ACPI 固定硬件接口，而是使用对应的软件替代方案。使用精简硬件 ACPI 的系统必须实现版本 5 或以上的 FACP 描述表（即 FADT 表），并且设置 HW_REDUCED_ACPI 标志位。这种方案称为功能型固定硬件（FFH），飞腾处理器遵循《ARM Functional Fixed Hardware Specification》规范要求。

根据 Linux 文档说明，对于 ARM64 上的 ACPI，表格分为如下类型：

- 必需：DSDT、FADT、GTDT、IORT、MADT、MCFG、RSDP、SPCR、XSDT
- 推荐：BERT、EINJ、ERST、HEST、HMAT、PCCT、SLIT、SRAT、SSDT
- 可选：BGRT、CPEP、CSRT、DBG2、DRTM、ECDT、FACS、FPDT、MCHI、MPST、MSCT、NFIT、PMTT、RASF、SBST、SPMI、STAO、TCPA、TPM2、UEFI、XENV
- 不支持：BOOT、DBGP、DMAR、ETDT、HPET、IBFT、IVRS、LPIT、MSDM、OEMx、PSDT、RSDT、SLIC、WAET、WDAT、WDRT、WPBT

飞腾平台要求实现上述必需表格内容，其它表格可由固件和 OS 根据需要实现。为支持 PCI 的 MSI 中断方式，还必须提供 IORT 表。

4.1 DSDT

DSDT 表是系统描述的一部分，DSDT 由一个系统描述表头和定义块（Definition Block）数据组成，这些定义块与其他定义块类似，不同在于这些定义块不能被卸载。定义块是一组 AML 对象，用于描述硬件实现细节信息。操作系统通过解析 DSDT 以及 SSDT（如果存在）中的定义块信息，获取设备信息，生成一个树结构的 ACPI 名字空间，该名字空间描述了系统中的硬件设备。OS 根据 ACPI 名字空间信息（如设备的_HID、_CID），加载相关设备驱动。

在飞腾系统中，处理器特有的 SOC 设备需要在 DSDT 表中描述。关于 SOC 设备的 ACPI 描述方法，参见第 5 章。

4.2 FADT

FADT 表是固定 ACPI 描述表，该表定义了对 ACPI 兼容操作系统来说，至关重要的固

定硬件 ACPI 信息，如一些 ACPI 硬件寄存器块的基址。FADT 还包含指向 DSDT 的指针。

对于 ARM64 来说，必需设置该表中的 HW_REDUCED_ACPI 标志。当该标志设置时，将忽略 ACPI 硬件寄存器接口相关的域,这些域应设置为 0。具体包括该表格偏移 46 到 108、偏移 148 到 232 的域，以及 FADT 标志位中的第 1、2、3、7、8、13、14、16 和 17 位

如果 FADT 中相关域包含 32 位和 64 位值，则优先使用 64 位值（如果 64 位值不为 0）。FADT 表中的 Preferred_PM_Profile 字段用来区分台式机、笔记本、服务器。如下表所示：

表 4-1 Preferred PM Profile 字段取值及含义

Preferred PM Profile 字段取值	取值含义
1	Desktop（台式机）
2	Mobile（笔记本）
4	Enterprise Server（服务器）

4.3 GTDT

通用定时器描述表 GTDT 描述系统通用定时器配置信息。通用定时器（GT）是基于 ARM 处理器的系统实现的标准定时器接口。GTDT 提供了系统通用定时器的中断配置，包括 per-processor 定时器、平台（内存映射）定时器。

GT 规范定义了下列 per-processor 定时器：

- 安全级别 1（EL1）定时器
- 非安全 EL1 定时器
- 非安全级别 2 EL2 定时器
- 虚拟定时器

以及下列平台（内存映射）定时器：

- GT 块
- SBSA 通用 Watchdog

飞腾处理器遵循上述规范。具体定时器配置信息（实现的定时器类型、中断、Always-on 能力、内存地址等）参见相关型号处理器的说明文档。如何描述这些配置，参见 ACPI 规范[1]5.2.24 节。

4.4 MADT

MADT 表即多 APIC 描述表，用于描述系统的中断控制器信息。对于飞腾系统，采用

GIC 中断控制器，因此在该表中需要使用 GIC 中断控制器结构（类型为 0xA-0xF），分别描述 GIC CPU 接口、GIC Distributor、GIC Redistributor、GIC ITS 信息。具体描述方法参见 ACPI 规范 5.2.12 节。

4.5 MCFG

MCFG 表即 PCI Express 内存映射地址空间基地址描述表。对于支持 PCI/PCIe 的平台，MCFG 表是必需的。参见《PCI Firmware Specification》。

OS 通过 ACPI 的 MCFG 表获得 PCI Express 的 ECAM 的基址。MCFG 表描述了非热插拔的 PCI 段组（Segment Group）的内存映射配置空间基址，或系统启动时可用的 PCI 段组的基址范围。该表格描述的地址范围对当前启动是非重定位的、非热插拔的。

_BBN: Boot Bus Numer 引导总线号。在多主桥系统中，_BBN 方法用于为某一特定总线提供 PCI_Config 操作区访问

_PRT：用于描述 PCI 中断路由：即 PCI INTx 中断到中断控制器的路由。_PRT 对象对于所有 PCI 主桥是必须的，用于提供 PCI INTx 路由信息。

_OSC：对于 PCI Express 主桥，该接口是必须的。

还需在 DSDT 或 SSDT 表中，在_SB 范围下声明一个设备，保留 ECAM 占据的内存空间，描述方式如下，注意设备的 HID 为“PNP0C02”。

```
// reserve ECAM memory range
  Device(RES0)
  {
      Name(_HID, EISAID("PNP0C02"))
      Name(_UID, 0)
      Name(_CRS, ResourceTemplate() {

        QWordMemory (ResourceConsumer, PosDecode, MinFixed, MaxFixed, Cacheable, ReadWrite,
            0x0000000000000000, // Granularity
            0x0000080040000000, // Range Minimum
            0x000008004FFFFFFF, // Range Maximum
            0, // Translation Offset
            0x10000000, // Length
            ,,)
    })
  }
```

4.6 RSDP

根系统描述指针 RSDP 用于定位根系统描述表 RSDT（32 位）或扩展系统描述表 XSDT（64 位）。对于 ARM64，该指针结构是必须的，由系统固件提供给 OS。

在 UEFI 系统中，EFI 系统表内有一个指向 RSDP 结构的指针。OS 加载器执行时，会获得 EFI 系统表指针，然后从中获得 RSDP 结构指针，并传递给 OSPM。

OS 加载器通过检查 EFI 系统表中的 EFI 配置表，定位出 RSDP 结构的指针。EFI 配置表表项包括一个 GUID/表格指针对。UEFI 为 APCI 定义了两个 GUID，一个用于 ACPI 1.0，另一个用于 ACPI 2.0 及后续版本。

ACPI 1.0 规范的 RSDP 结构的 EFI GUID 为：

EB9D2D30-2D88-11D3-9A16-0090273FC14D

ACPI 2.0 及以后规范的 RSDP 结构的 EFI GUID 为：

8868E871-E4F1-11D3-BC22-0080C73C8881

OS 加载器将首先使用当前版本的 GUID 查找 RSDP 结构指针，如果找到，则使用对应指针；如果没找到，将使用 ACPI 1.0 版本的 GUID 查找。

4.7 SPCR

如果支持内核启动时不带 console=<device>参数，则需要提供 SPCR 表。对于 Linux 系统，该表提供 earlycon 控制台的配置信息。

4.8 XSDT

扩展系统描述表 XSDT 功能和根系统描述表 RSDT 相同，但描述头中使用的物理地址为 64 位。RSDP 结构可以指向 RSDT 和 XSDT。如果存在 XSDT，则使用 XSDT。ACPI 系统通过根系统描述表获取其它 ACPI 表格（如 DSDT）的物理地址。Phytium 处理器为 64 位处理器，因此使用该表，而不是 RSDT。

4.9 IORT

为支持 PCI 的 MSI/MSI-X 中断方式，需提供 IORT 表（输入输出重映射表）。参见《IO Remapping Table Platform Design Document》。IORT 表描述了基于 ARM 的系统的 IO 拓扑。具体来说，IORT 提供下列描述：

- 提供 IO 拓扑、SMMU 和 GIC ITS 的 ACPI 描述
- 标识位于 SMMU 后的组件
- 标识位于 ITS 或 ITS 组后的组件
- 描述 PCIe Root Complex 与 MCFG 表和 ACPI 名字空间的联系
- 描述 ACPI 名字空间中设备的 IO 关系

使用 IORT 结构描述 GIC ITS 与 PCIe 控制器之间的关联关系。

4.10 PPTT

PPTT 表全称为 Processor Properties Topology Table ，描述了处理器的拓扑信息以及共享的资源（caches）。具体描述方法参见 ACPI 规范 5.2.29 节。PPTT 提供下列描述：

- 提供处理器 core 的拓扑结构
- 描述了处理器 core 与 caches 间的关系
- 描述了 caches 的类型
- 描述了芯片厂商 ID、芯片家族 ID、芯片 ID

以腾云 S2500 双路为例。双路 S2500 包含 2 个 CPU，每个 CPU 包含 8 个 panels，panel 内的处理器 core 共享一个 8MB L3 cache，每个 panel 包含 2 个 clusters，cluster 内的处理器 core 共享一个 2MB L2 cache，每个 cluster 包含的 4 个 cores，每个 core 包含一个 32KB L1 指令 cache 和一个 32KB L1 数据 cache。S2500 PPTT 表描述如下：

表 4-2 PPTT Table

Field	Byte Length	Byte Offset	Value
Header			
Signature	4	0	"PPTT"
Table Length	4	4	0x0000BDD4
Revision	1	8	0x01
Checksum	1	9	0xD5
OEM ID	6	10	"FT-LTD"
OEM Table ID	8	16	"PHYTIUM."

OEM Revision	4	24	20180509
.....			
Subtable Type	1	36	00[Processor]
……			
Physical package	1	40	1
……			
Subtable Type	1	60	00[Processor]
……			
Physical package	1	64	0
……			
Subtable Type	1	84	01[cache]
……			
Size	4	96	0x800000
……			
Subtable Type	1	108	00[Processor]
……			
Physical package	1	112	0
……			
Subtable Type	1	132	01[cache]
……			
Size	4	144	0x200000
……			
Subtable Type	1	156	00[Processor]
……			
Physical package	1	160	0
……			
ACPI Processor ID	4	168	0
……			
Subtable Type	1	184	01[cache]
……			
Size	4	196	0x8000
……			
Subtable Type	1	208	01[cache]
……			
Size	4	220	0x8000
……			
Subtable Type	1	156	00[Processor]
……			
Physical package	1	160	0
……			
ACPI Processor ID	4	168	1
……			
Subtable Type	1	184	01[cache]

……			
Size	4	196	0x8000
……			
Subtable Type	1	208	01[cache]
……			
Size	4	220	0x8000
……			
Subtable Type	1		02[ID]
……			
VENDOR_ID	4		54594850[phytium]
LEVEL_1_ID	8		4[S2500]
LEVEL_2_ID	8		0[S2500/64]
……			

5 飞腾系统设备 ACPI 描述

5.1 概述

飞腾系列处理器采用 ARMv8 指令集。对于支持 ACPI 的固件设计者，需要关注的飞腾处理器模块有：

1) CPU 核模块：处理器中 CPU 核的数目、层次结构、电源状态相关描述。
2) PCIE 模块：PCIE 主桥数目、配置等描述。
3) 其它 SOC 外设。主要分为：
   - 快速 IO 模块：主要包括 SD 卡控制器模块 SDC、HDAudio 控制器
   - 慢速 IO 模块：主要包括 UART 串口、LPC、GPIO、I2C、QSPI、通用 SPI、RTC 等设备。
   - GMAC 模块：以太网控制器。

具体信息参见相应型号处理器的参考文档。

5.2 电源域划分

根据处理器型号不同，芯片对电源域的划分有所区别。为了实现更细粒度的功耗控制，飞腾处理器芯片内部一般实现了多个细粒度的电源域划分，各个电源域可分别关断。对电源域进行关断，一般需要先配置对应寄存器，然后执行 WFI 指令。对电源域的具体操作，可由固件的 PSCI 实现，也可由 ACPI 的控制方法实现。

5.3 处理器低功耗状态

在 ACPI 中，可使用处理器和处理器容器设备描述处理器层次，将系统中所有处理器以分层的方式描述成一个树结构。处理器容器描述一组关联的处理器；当处理器以某种方式关联，如共享 cache 或被一共同低功耗模式影响，则称该处理器属于某一容器。

在处理器层次中，每个节点拥有该节点所特有的低功耗状态。ACPI 称该电源状态为局部电源状态（Local Power State）。例如，处于最底层的 CPU 节点，可能包括的局部电源状态有：clock gate、retention 和 power down。该局部电源状态使用_LPI 对象描述。

当运行在某一处理器上的 OS 检测到该处理器没有更多工作调度时，需要选择一个低功

耗空闲状态。该状态可能不仅仅影响该处理器自身。例如，一个进入空闲状态的处理器可能是系统或处理器容器中的最后一个活动处理器，因此系统可以选择一种影响多个处理器的电源状态。为选择这样一个状态，OS 需要为处理器层次中每个受影响的级别选择一个局部电源状态。

然而，大多数硬件体系结构仅支持从 OS 到平台的单个电源状态请求；即，不能为每个层次节点发起单独的电源状态请求。因此，OS 必须组合每个层次的局部状态形成单个的组合电源状态（Composite Power State）。平台根据组合电源状态请求进行操作。

空闲状态协调：当某一处理器请求进入某一低功耗状态时，可能会影响多个处理器或处理器电源层次的电源状态，这要求协调处理器的空闲状态请求。APCI 支持两种协调机制：平台协调、OS 发起。

飞腾系统采用平台协调方式，在这种方式中，由硬件平台负责跨处理器的空闲状态协调。OSPM 针对所有处理器层次发出一个请求，该请求蕴含着每个处理器为其自身、父节点、父节点的父节点等的局部功耗状态的投票选择，平台固件综合所有处理器的选择，最终选择一个满足约束的组合目标电源状态，并将平台置于该状态。

关于处理器的低功耗状态描述和电源管理，飞腾系统遵循《ARM Functional Fixed Hardware Specification》。ACPI 表中对处理器_LPI 的描述，需遵循该规范；由平台固件实现底层硬件的具体操作，通过 PSCI 接口供 OS 等上层软件调用。

5.4 PCIE 控制器

对于 UEFI 等固件实现来说，PCIE 桥控制器信息要通过飞腾 Base Firmware 接口规范提供的 PCI_HOST_BRIDGE 接口动态获取。参见《Phytium Base Firmware 接口规范》。PCIe 桥控制器配置描述方法参考《PCI Firmware Specification》。

5.5 处理器 SOC 设备

需遵循 APCI 规范[1]中设备相关描述标准。

飞腾系统的 SOC 设备需要描述的内容有：

- **硬件标志 HID**：对于第三方设备，使用第三方定义的 ACPI HID。对于飞腾的自研设备，使用飞腾定义的 HID，其中，前 4 个字符为“PHYT”[8]。

注：本文历史版本 v1.0 中 HID 定义与当前版本有所不同。如果需要兼容 v1.0 规范，则

需要在设备描述时增加“_CID”属性，对应的值设置为 v1.0 中的 HID 值。

表 5-1 Phytium Device HID 列表

设备	HID	CID
GPIO	PHYT0001	*FTGP0001*
RTC	PHYT0002	
I2C	PHYT0003	*FTI20001*
GMAC	PHYT0004	*FTGM0001*
SDC	PHYT0005	*FTSD0001*
HDAudio	PHYT0006	*FTHD0001*
LPC	PHYT0007	*LPC0001*
SCPI	PHYT0008	*FTSC0001*
MailBox	PHYT0009	*FTMB0001*
CAN	PHYT000A	
EC	PHYT000B	*FTEC0001*
Battery	PHYT000C	*FTEC0002*
Sensor	PHYT000D	*FTEC0003*
SPI	PHYT000E	
SCMI	PHYT0010	
SCPI-clock	PHYT8001	*FTCK0001*
Fixed-Clock	PHYT8002	*FTCK0002*
Trusted OS	PHYT8003	

- **内存资源**：设备在内存映射地址空间中占据的资源范围，具体信息参考相关型号处理器的说明文档。
- **中断资源**：系统设计时为设备分配的硬件中断号，具体信息参考相关型号处理器的说明文档。
- **设备电源状态**：设备支持的电源状态。
- **其它**：其它与设备相关的操作和属性。设备操作使用相关 ACPI 控制方法描述，设备专有属性使用_DSD 方法描述，具体信息根据设备类型确定。

处理器 SOC 设备的资源需求，可参见对应的软件编程手册。比如，处理器飞腾 FT-2000/4

的相关 SOC 设备资源信息，即可通过查阅参考文献[7]获得。

附录 A SOC 设备的 ACPI 描述

本文介绍飞腾 SOC 设备的 APCI 描述。系统 SOC 设备的具体资源信息，与处理器实现相关，需要从对应的软件编程手册[7]获得。

A.1 UART

兼容 ARM 协议规范的 PL011，对应的“_HID”为“ARMH0011”。

对应的 Linux 驱动为：amba-pl011.c，该驱动支持 ACPI，无需修改。

SOC 可能包含多个 UART[7]，UART ACPI 描述示例：

```
Scope(_SB)
{
    //UART 1
    Device(UAR1) {
        Name(_HID, "ARMH0011")
        Name(_UID, 1)
        Name(_CRS, ResourceTemplate() {
          Memory32Fixed(ReadWrite, 0x28001000, 0x1000)
          Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) {39}
        })

        Method (_STA, 0, NotSerialized) {
          Return(0x0F)
        }
    }

 // UART 0
 Device(UAR0) {
        Name(_HID, "ARMH0011")
        Name(_UID, 0)
        Name(_CRS, ResourceTemplate() {
          Memory32Fixed(ReadWrite, 0x28000000, 0x1000)
          Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) { 38 }
        })

        Method (_STA, 0, NotSerialized) {
        Return(0x0F)
        }
    }

    //UART 2
```

```
        Device(UAR2) {
          Name(_HID, "ARMH0011")
          Name(_UID, 2)
          Name(_CRS, ResourceTemplate() {
            Memory32Fixed(ReadWrite, 0x28002000, 0x1000)
            Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) {40}
          })

          Method (_STA, 0, NotSerialized) {
          Return(0x0F)
          }
        }

    //UART 3
    Device(UAR3) {
          Name(_HID, "ARMH0011")
          Name(_UID, 3)
          Name(_CRS, ResourceTemplate() {
            Memory32Fixed(ReadWrite, 0x28003000, 0x1000)
            Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) {41}
          })

          Method (_STA, 0, NotSerialized) {
          Return(0x0F)
          }
        }
}
```

A.2 Fixed-clock

为了兼容某些设备的驱动框架，便于驱动获得时钟频率相关的属性，需要定义一个伪设备 fixed-clock。

```
Scope(_SB)
{
    Device (CLK3) {
      Name (_HID, "PHYT8002")
      Name (_CID, "FTCK0002")       /* compatible with v1.0 */
      Name (_UID, 0x0)
      Name (_DSD, Package() {
        ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
          Package () {
            Package(2) {"clock-frequency", 600000000}
          }
        })
```

```
        Method (FREQ, 0x0, NotSerialized) {
            Return (600000000)
        }
    }
}
```

A.3 GPIO

飞腾处理器可能包含多个 GPIO 模块，每个模块又可能分成多个组，每组又有多个 GPIO[7]。

ACPI 描述示例：

```
Scope(_SB)
{
 Device(GPI0) {
        Name(_HID, "PHYT0001")
        Name(_CID, "FTGP0001")    /* compatible with v1.0 */
        Name(_UID, 0)
        Name (_CRS, ResourceTemplate ()    {
          Memory32Fixed (ReadWrite, 0x28004000, 0x1000)
          Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) { 42 }
        })

        Device(GP00) {
          Name(_UID,0)
          Name (_DSD, Package () {
            ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
            Package () {
              Package () {"reg",0},
              Package () {"snps,nr-gpios",8},
              Package () {"nr-gpios",8},
          }
          })
        }
    }
    Device(GPI1) {
      Name(_HID, "PHYT0001")
      Name(_CID, "FTGP0001")    /* compatible with v1.0 */
      Name(_UID, 1)
      Name (_CRS, ResourceTemplate ()    {
        Memory32Fixed (ReadWrite, 0x28005000, 0x1000)
        Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) { 43 }
      })
```

```
        Device(GP00) {
          Name(_UID, 0)
          Name (_DSD, Package () {
            ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
            Package () {
              Package () {"reg",0},
              Package () {"snps,nr-gpios",8},
              Package () {"nr-gpios",8},
            }
          })
        }
      }
}
```

A.4 I2C

SOC 可能包含多个 I2C 模块[7]，不同 I2C 下可能挂载不同的设备。

ACPI 描述示例如下。其中，假定 I2C0 挂载了 EEPROM，I2C1 挂载了 RTC DS1339：

```
Scope(_SB)
{
    Device (CLK2) {
      Name (_HID, "PHYT8002")
      Name (_CID, "FTCK0002")    /* compatible with v1.0 */
      Name (_UID, 0x01)
      Name (_DSD, Package() {
        ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
          Package () {
            Package(2) {"clock-frequency", 48000000}
          }
        })

      Method (FREQ, 0x0, NotSerialized) {
        Return (48000000)
      }
    }

    Device (I2C0) {
      Name (_HID, "PHYT0003")
      Name (_CID, "FTI20001")    /* compatible with v1.0 */
      Name (_UID, Zero)
      Name (_CRS, ResourceTemplate () {
        Memory32Fixed (ReadWrite, 0x28006000, 0x1000)
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 44 }
```

```
        })
        Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
          Package () {
            Package () {"clock-frequency", 0x186a0},
            Package () {"clocks", Package () {"\\_SB.CLK2"}}
        }
        })

    Device (EPR0) {
          Name (_HID, "INT0002")
          Name (_UID, Zero)
          Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
            Package () {
              Package () {"pagesize", 16}
            }
          })

          Name (_CRS, ResourceTemplate () {
            I2cSerialBus (0x0057, ControllerInitiated, 400000, AddressingMode7Bit, "\\_SB.I2C0", 0x00,
ResourceConsumer, ,)
          })
        }

      Device (I2C1) {
        Name (_HID, "PHYT0003")
        Name (_CID, "FTI20001")   /* compatible with v1.0 */
        Name (_UID, 1)
        Name (_CRS, ResourceTemplate () {
          Memory32Fixed (ReadWrite, 0x28007000, 0x1000)
          Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 45 }
        })
        Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
          Package () {
            Package () {"clock-frequency", 0x186a0},
            Package () {"clocks", Package () {"\\_SB.CLK2"}}
          }
        })

        Device (RTC0) {
          /* ASL 编译器可能因 6 个字符长度的_HID 报错，_HID 填一个 OS 不能识别的、格式满足规
            范要求的任意字符串即可。
```

```
                */
                Name (_HID, "FTDS1339")
                Name (_CID, "DS1339")     /* DS1339 RTC, Linux 驱动匹配的 ID */
                Name (_UID, Zero)
                Name (_CRS, ResourceTemplate () {
                  I2CSerialBusV2  (0x68,  ControllerInitiated,  100000,AddressingMode7Bit,  "\\_SB.I2C1",  0,
ResourceConsumer, ,)
                })
              }
            }
        }
```

A.5 Watchdog

飞腾平台的 Watchdog 遵循 ARM SBSA 标准，对应的 Linux 驱动为：

- drivers/acpi/arm64/gtdt.c
- drivers/watchdog/sbsa_gwdt.c

gtdt.c 驱动从 GTDT 表获取 watchdog 信息，初始化 watchdog platform device 信息。sbsa_gwdt.c 驱动为设备分配资源、初始化并对设备进行操作。具体实现参考源代码。

Watchdog 需要在 GTDT 表中进行描述。需要提供的参数有：

- Refresh Frame 物理地址
- Watchdog ControlFrame 物理地址
- Watchdog Timer GSIV：SBSA 通用 watchdog timer 使用的全局中断号
- Watchdog Timer Flags: 标志位，意义如下表所示。

表 附 A-0-1 Watchdog Timer Flags

位 0	Timer 中断模式	1：边沿触发 0：电平触发
位 1	Timer 中断 polarity	1：低有效 0：高有效
位 2	安全 Timer	1：timer 是安全的 0：timer 是非安全的
其它位	保留	必须为 0

具体资源需求参见处理器软件编程手册[7]。

A.6 GMAC

SOC 可能包含多个 GMAC 模块[7]。

ACPI 描述示例如下，以 2 个 GMAC 为例：

```
Scope(_SB)
{
    Device (ETH0) {
      Name (_HID, "PHYT0004")
      Name (_CID, "FTGM0001")          /* compatible with v1.0 */
      Name (_UID, 0)
      Name (_CCA, 1)
      Name (_CRS, ResourceTemplate () {
        Memory32Fixed (ReadWrite, 0x2820c000, 0x2000)
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 81 }
      })
      Name (_DSD, Package () {
        ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
        Package () {
          Package () {"interrupt-names", "macirq"},
          Package () {"clocks", 3},
          Package () {"clock-names", "stmmaceth"},
          Package () {"snps,pbl", 0x10},
          Package () {"snps,abl", 0x20},
          Package () {"snps,burst_len", 0x0e},
          Package () {"snps,multicast-filter-bins", 0x40},
          Package () {"snps,perfect-filter-entries", 0x41},
          Package () {"max-frame-size", 0x2328},
          Package () {"phy-mode", "rgmii-rxid"},
          Package () {"clock-frequency",250000000},
          Package () {"bus_id",0},
          Package () {"txc-skew-ps", 0x3e8},
          Package () {"rxc-skew-ps", 0x3e8},
          Package () {"mdc_clock_selection", 0x5},          /* CSR_250_300M, MDC = clk_scr_i/122 */
        }
      })

      Method (_STA, 0, NotSerialized) {
      Return(0x0F)
      }
    }

  Device (ETH1) {
      Name (_HID, "PHYT0004")
      Name (_CID, "FTGM0001")          /* compatible with v1.0 */
      Name (_UID, 1)
```

```
            Name (_CCA, 1)
            Name (_CRS, ResourceTemplate () {
              Memory32Fixed (ReadWrite, 0x28210000, 0x2000)
              Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 82 }
            })
            Name (_DSD, Package () {
              ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
              Package () {
                Package () {"interrupt-names", "macirq"},
                Package () {"clocks", 3},
                Package () {"clock-names", "stmmaceth"},
                Package () {"snps,pbl", 0x10},
                Package () {"snps,abl", 0x20},
                Package () {"snps,burst_len", 0x0e},
                Package () {"snps,multicast-filter-bins", 0x40},
                Package () {"snps,perfect-filter-entries", 0x41},
                Package () {"max-frame-size", 0x2328},
                Package () {"phy-mode", "rgmii-rxid"},
                Package () {"clock-frequency",250000000},
                Package () {"bus_id",1},
                Package () {"txc-skew-ps", 0x3e8},
                Package () {"rxc-skew-ps", 0x3e8},
              }
            })

            Method (_STA, 0, NotSerialized) {
              Return(0x0F)
            }
        }
    }
```

A.7 SDC

飞腾处理器内置 SDC 的接口细节可以查阅相关手册[7]。

ACPI 描述示例：

```
Scope(_SB)
{
    Device (SDC0) {
      Name (_HID, "PHYT0005")
      Name (_CID, "FTSD0001")          /* compatible with v1.0 */
      Name (_UID, 0)
      Name (_CCA, 1)
      Name (_CRS, ResourceTemplate () {
        Memory32Fixed (ReadWrite, 0x28207C00, 0x100)
```

```
            Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 52,53,54}
        })

        Method (_STA, 0, NotSerialized) {
        Return(0x0F)
        }
 }

}
```

A.8 HDAudio

飞腾处理器内置 HDA 的接口细节可以查阅相关手册[7]。

ACPI 描述示例：

```
Scope(_SB)
{
    Device (HDA0) {
      Name (_HID, "PHYT0006")
      Name (_CID, "FTHD0001")          /* compatible with v1.0 */
      Name (_UID, 0)
      Name (_CCA, 1)
      Name (_CRS, ResourceTemplate () {
        Memory32Fixed (ReadWrite, 0x28206000, 0x1000)
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 55}
      })

      Method (_STA, 0, NotSerialized) {
      Return(0x0F)
      }
 }
}
```

A.9 LPC

飞腾处理器内置 LPC 接口细节可以查阅相关手册[7]。

ACPI 描述示例：

```
Scope(_SB)
{
  Device (LPC0) {
    Name (_HID, "PHYT0007")
    Name (_CID, "LPC0001")          /* compatible with v1.0 */
```

```
        Name (_UID, Zero)
        Name (_CRS, ResourceTemplate () {
          Memory32Fixed (ReadWrite, 0x20000000, 0x08000000,)
          Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive, ,, ) { 0x25 }
        })
      }
    }
```

A.10 SCPI

飞腾处理器内置系统控制模块，其操作接口可能遵循 SCPI 协议，细节可以查阅相关手册[7]。

SCPI 协议涉及多个操作接口，如 mailbox、调频使用的 scpi-clock 等。

```
Scope(_SB)
{
    Device(CLK1){
    Name (_HID, "PHYT8001")       /* scpi-clock */
    Name (_CID, "FTCK0001")       /* compatible with v1.0 */
    Name(_UID,0)
    Name (_DSD, Package () {
      ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
      Package () {
        Package () {"compatible", "arm,scpi-clocks"}
      }
    })

      Device(CK10) {
        Name(_ADR,0)
        Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
          Package () {
            Package () {"compatible", "arm,scpi-dvfs-clocks"},
            Package () {"#clock-cells", 1},
            Package () {"clock-indices", Package () {0,1}},
            Package () {"clock-output-names", Package () {"c0","c1"}}
          }
        })
      }
    }

    Device (SCPI) {
      Name (_HID, "PHYT0008")        /* SCPI */
      Name (_CID, "FTSC0001")        /* compatible with v1.0 */
```

```
        Name (_UID, 0)
        Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
          Package () {
            Package () {"mbox", \_SB.MBX0},
            Package () {"shmem_start", 0x2a007000},
            Package () {"shmem_size", 0x800}
          }
        })

    Device(SEN1){
      Name(_HID, "PHYT000D")
      Name(_CID, "FTSS0001")
      Name (_DSD, Package () {
        ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
        Package () {
          Package () {"compatible", "arm,scpi-sensors"},
          Package () {"#thermal-sensor-cells", 1}
        }
      })
    }
  }

    //Mailbox
    Device(MBX0){
      Name (_HID, "PHYT0009") /* MailBox */
      Name (_CID, "FTMB0001") /* compatible with v1.0 */
      Name (_UID, 0)
      Name (_CRS, ResourceTemplate () {
        Memory32Fixed (ReadWrite, 0x2A000000, 0x1000)
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 80 }
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 79 }
      })
    }
}
```

为使能 scpi，还需在 CPU 描述中增加使用的时钟，描述如下：

```
  Device(CPU0) { //    Cluster 0, Cpu 0
        Name(_HID, "ACPI0007")
        Name(_UID, 0)
        Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
         Package () {
                Package () {"clock-name","c0"},
                Package () {"clock-domain",0},
```

```
            }
            })
    }
```

其中，_DSD 中增加了 clock-name 和 clock-domain 域，用于指定本 CPU 核使用的调频时钟，其值对应到 SCPI 描述中的 CLK10 设备的相关值。

A.11 CAN

SOC 可能包含多个 CAN 接口[7]。

CAN 的 ACPI 描述示例如下，以包含 3 个 CAN 为例。

```
Scope(_SB)
{
/* CLK3 建议删除，本文保留为了兼容 v1.1*/
    Device (CLK3) {
      Name (_HID, "PHYT8002")          /* Fixed-Clock */
      Name (_CID, "FTCK0002")          /* compatible with v1.0 */
      Name (_UID, 0x0)
      Name (_DSD, Package() {
        ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
          Package () {
            Package(2) {"clock-frequency", 600000000}
          }
        })

      Method (FREQ, 0x0, NotSerialized) {
        Return (600000000)
      }
    }

    // can0
    Device(CAN0) {
      Name (_HID, "PHYT000A")
      Name (_CID, "FTCN0001")          /* compatible with v1.0 */
      Name(_UID, 0)
      Name(_CRS, ResourceTemplate() {
        Memory32Fixed(ReadWrite, 0x28207000, 0x400)
        Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) {119}
      })

      Name (_DSD, Package () {
      ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
      Package () {
        Package () {"clocks", Package (){"\\_SB.CLK3"}}, /*  建议删除  */
```

```
        Package () {"clock-frequency", 600000000},
        Package () {"clock-names", "ftcan_clk"},
        Package () {"tx-fifo-depth", 0x40},
        Package () {"rx-fifo-depth", 0x40}
    }
})

    Method (_STA, 0, NotSerialized) {
        Return(0x0F)
    }
}

// can 1
Device(CAN1) {
    Name (_HID, "PHYT000A")
    Name (_CID, "FTCN0001")          /* compatible with v1.0 */
    Name(_UID, 1)
    Name(_CRS, ResourceTemplate() {
        Memory32Fixed(ReadWrite, 0x28207400, 0x400)
        Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) {123}
    })

    Name (_DSD, Package () {
    ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
    Package () {
        Package () {"clocks", Package (){"\\_SB.CLK3"}}, /*  建议删除*/
        Package () {"clock-frequency", 600000000},
        Package () {"clock-names", "ftcan_clk"},
        Package () {"tx-fifo-depth", 0x40},
        Package () {"rx-fifo-depth", 0x40}
    }
})

    Method (_STA, 0, NotSerialized) {
        Return(0x0F)
    }
}

// can 2
Device(CAN2) {
    Name (_HID, "PHYT000A")
    Name (_CID, "FTCN0001")          /* compatible with v1.0 */
    Name(_UID, 2)
    Name(_CRS, ResourceTemplate() {
```

```
            Memory32Fixed(ReadWrite, 0x28207800, 0x400)
            Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) {124}
        })

        Name (_DSD, Package () {
        ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
        Package () {
            Package () {"clocks", Package (){"\\_SB.CLK3"}},/*  建议删除*/
            Package () {"clock-frequency", 600000000},
            Package () {"clock-names", "ftcan_clk"},
            Package () {"tx-fifo-depth", 0x40},
            Package () {"rx-fifo-depth", 0x40}
        }
    })

        Method (_STA, 0, NotSerialized) {
            Return(0x0F)
        }
    }
}
```

A.12 Trusted OS

飞腾处理器支持 Secure 和 Non-Secure 两种模式，Secure 模式下运行相应的 Trusted OS。以 OP-TEE 为例，ACPI 描述如下：

```
Scope(_SB)
{
    Device (OPTE) {
        Name (_HID, "PHYT8003")
        Name (_CID, "FTOP0001")
        Name (_UID, 0)
        Name (_DSD, Package () {
            ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
            Package () {
                Package () {"compatible", "linaro,optee-tz"},
                Package () {"method", "smc"},
            }
        })
 }
}
```

A.13 SPI

飞腾处理器内置 SPI 接口，接口的详细信息见处理器相关手册[7]。

ACPI 描述如下：

```
Scope(_SB)
{
    Device(SPI0) {
      Name(_HID, "PHTY000E")
      Name(_UID, 0)
      Name(_CRS, ResourceTemplate() {
        Memory32Fixed(ReadWrite, 0x2800c000, 0x1000)
        Interrupt(ResourceConsumer, Level, ActiveHigh, Exclusive) {50}
      })

      /*  以 SPI 下挂 Linux SPT  设备为例  */
      Device(FLAS) {
        Name(_HID, "SPT0001")
        Name(_UID, 0)
        Name (_DSD, Package() {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
            Package () {
              Package () {"reg",0},
              Package() {"spi-max-frequency", 48000000}
            }
        })
       Name(_CRS, ResourceTemplate () {
         // Index 0
            SPISerialBus (              // SCKL - GPIO 11 - Pin 23
                                        // MOSI - GPIO 10 - Pin 19
                                        // MISO - GPIO 9    - Pin 21
                                        // CE0    - GPIO 8    - Pin 24
              0,                        // Device selection (CE0)
              PolarityLow,              // Device selection polarity
              FourWireMode,             // WireMode
              8,                        // DataBit len
              ControllerInitiated,      // Slave mode
              4000000,                  // Connection speed
              ClockPolarityLow,         // Clock polarity
              ClockPhaseFirst,          // Clock phase
              "\\_SB.SPI0",             // ResourceSource: SPI bus controller name
              0,                        // ResourceSourceIndex
                                        // Resource usage
                                        // DescriptorName: creates name for offset of resource descriptor
            )                           // Vendor Data
         })
```

```
        }
    }
}
```

A.14 SCMI

飞腾处理器可能内置系统控制模块，其操作接口可能遵循 SCMI 协议。

SCMI 协议还涉及 mailbox 操作接口。

ACPI 描述示例如下：

```
Scope(_SB) {

 // Mailbox
 Device(MBX0){
    Name (_HID, "PHYT0009")
    Name (_CID, "FTMB0001") /* compatible with v1.0 */
    Name (_UID, 0)
    Name (_CRS, ResourceTemplate () {
        Memory32Fixed (ReadWrite, 0x2A000000, 0x1000)
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 80 }
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) { 79 }
    })
 } // end of Device(MBX0)

     Device (SCMI) {
         Name (_HID, "PHYT0010")
         Name (_UID, 0)
         Name (_DSD, Package () {
           ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
              Package () {
                Package () {"mbox", \_SB.MBX0},
                Package () {"shmem_start", 0x2a007000},
                Package () {"shmem_size", 0x800}
              }
         })

          Device(POWR){
                Name(_ADR,0)
                Name (_DSD, Package () {
                        ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
                        Package () {
                                Package () {"reg", 0x11},
                                Package () {"#power-domain-cells", 1}
```

```
                    }
                })
            }

        Device(DVFS){
            Name(_ADR,1)
            Name (_DSD, Package () {
                    ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
                    Package () {
                            Package () {"reg", 0x13},
                            Package () {"#clock-cells", 1}
                    }
                })
            }

        Device(CLK){
            Name(_ADR,2)
            Name (_DSD, Package () {
                    ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
                    Package () {
                            Package () {"reg", 0x14},
                            Package () {"#clock-cells", 1}
                    }
                })
            }

        Device(SEN){
            Name(_ADR,3)
            Name (_DSD, Package () {
                    ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
                    Package () {
                            Package () {"reg", 0x15},
                            Package () {"#thermal-sensor-cells", 1}
                    }
                })
            }

    }   // end of   Device (SCMI)

} // end of   Scope(_SB)
```

A.15 PMU

飞腾处理器内置性能监控模块（PMU，Performance Monitor Unit）。

PMU 中断是 PPI 中断，在 MADT 表的 GICC 结构中描述[1]。其中，Performance Interrupt GSIV 表示 PMU 的中断号，Flags 字段的 Performance Interrupt Mode 位表示中断的触发模式。

A.16 Temperature Sensor

ACPI 规范[1]支持通过 Thermal Zone 来描述温度传感器。

本文 FT-2000+/64 SOC 内置传感器为例，将一个温度传感器定义成一个 ThermalZone，并将一个处理器核动态作为散热设备，散热机制是动态调频，温度查询相关的方法的返回值以开尔文温度的十分之一（K/10）为单位。示例中，SOC 温度传感器的寄存器中的温度值以千分之一摄氏度（mC）为单位，_M2K 实现不同单位温度值间的转换，将 mC 转换为 K/10。_C2K 将 C 转换为 K/10。

```
ThermalZone (TZ0) {
    Method(_PSV) { Return(_C2K(70)) }                    /* passive cooling temp */
    Method(_HOT) { Return(_C2K(85)) }                    /* hot temp */
    Method(_CRT) { Return(_C2K(95)) }                    /* critical temp */
    Method(_TMP) {                                       /* current temp */
      OperationRegion(TMP, SystemMemory, 0x80028780000, 0x1000)
      Field(TMP, DWordAcc, NoLock, Preserve) {
            A0, 32,
        Offset(8),
        D0, 32,
      }
      Store(0x30, A0)
      Store(0x301, D0)
      Store(0x201, D0)
      Store(0x601, D0)
      Store(0x201, D0)
      Store(0x110, A0)
      Sleep(1)
      Store(D0, Local0)
      And(Local0, 0xfff, Local1)
      Sleep(1)
      Store(39375, Local2)
      Subtract(Multiply(Local1, 49), Local2, Local0)
      Sleep(1)
      Return(_M2K(Local0))
    }   /* _TMP */
}   /* ThermalZone(TZ0) */

    Method(_M2K, 1, Serialized) {                /* mC to K/10*/
      Divide(Arg0, 100, Local2, Local1)
```

```
    Add(Local1, 0xAAC, Local0)
    If(LLessEqual(Local0, 0x0AAC))
    {
        Store(0x0C8C, Local0)
    }
    Else
    {
        If(LGreater(Local0, 0x0FAC))
        {
            Store(0xC8C, Local0)
        }
    }
    Return(Local0)
}

Method(_C2K, 1, Serialized) {                    /* C to K/10 */
    Add(Multiply(Arg0, 0x0A), 0xAAC, Local0)
    If(LLessEqual(Local0, 0x0AAC))
    {
        Store(0x0C8C, Local0)
    }
    Else
    {
        If(LGreater(Local0, 0x0FAC))
        {
            Store(0xC8C, Local0)
        }
    }
    Return(Local0)
}
```

A.17 IPMI

服务器系统中可能有带外管理控制器 BMC（Baseboard Management Controller），BMC 软件与处理器上的系统软件（比如，BIOS 或 OS）基于 IPMI 协议通信。

比如，BMC 挂载在飞腾 FT-2000+/64 处理器的 LPC 总线下，飞腾处理器是 LPC Master，BMC 端是 LPC Slave，BMC 与处理器基于 IPMI KCS 接口协议进行通信。

IPMI 的 ACPI 描述有两种方式：Name Space 和 SPMI（Server Platform Management Interface）表。但是，对于 non-PCI 接口的场景，推荐使用 ACPI Name Space 方式来描述。因此，本文也推荐用 Name Space 方式来描述 IPMI。

A.17.1 Name Space

```
Device (IPI0) {
        Name (_HID, "IPI0001")
        Name (_UID, 0)
        Name (_STR, Unicode("IPMI_KCS"))
        Name(_CCA, 1)
        Method (_IFT) {                 /* Interface Type*/
          Return (0x01)                 /* KCS */
        }
        Method (_SRV) {
          Return (0x0200)               // IPMI Spec Revision 2.0
        }
        Name (_CRS, ResourceTemplate () {       // _CRS: Current Resource Settings
          QWordMemory (                         // LPC memory-mapped registers region
            ResourceConsumer,
            PosDecode,
            MinFixed,
            MaxFixed,
            Cacheable,
            ReadWrite,
            0x0,                    // Granularity
            0x80020000ca2,      // Min Base Address
            0x80020000ca3,      // Max Base Address
            0x0,                    // Translate
            0x2                     // Length
          )
        })   /* _CRS*/
  }
```

A.17.2 SPMI

表 附 A-0-2 SPMI Table

Field	Byte Length	Byte Offset	Value
Header			
Signature	4	0	"SPMI"
Length	4	5	0x00000041
Revision	1	8	0x05
Checksum	1	9	0x24
OEMID	6	10	"PHYLTD"
OEM Table ID	8	16	"PHYTIUM."
OEM Revision	4	24	0x00000001
Creator ID	4	28	"PHYT"
Creator Revision	4	32	0x00000001
Interface Type	1	36	0x01
Reserved	1	37	0x01
Spec revision	2	38	0x0200
Interrupt Type	1	40	0x00
GPE	1	41	0x00
Reserved	1	42	0x00
PCI Device Flag	1	43	0x00
Global Interrupt	4	44	0x00000000
Base Address			
Space ID	1	48	0x00
Bit Width	1	49	0x08
Bit Offset	1	50	0x00
Access Width	1	51	0x01
Address	8	52	0x0000080020000CA2
PCI Segment	1	60	0x00
PCI Bus	1	61	0x00
PCI Device	1	62	0x00

PCI Function	1	63	0x00
Reserved	1	64	0x00

附录 B CPU 的 ACPI 描述

以 4 核处理器为例，假定处理器核分为 2 个 cluster，每个 cluster 包含 2 个处理器核。处理器采用 SCPI 接口调频进行动态功耗管理。为说明处理器所在的调频时钟域，在 ACPI 表中，增加了相关描述内容。CPU 描述示例如下，其中的 clock-name 和 clock-domain 描述了用于调频的 SCPI 时钟名和时钟域 ID，供 SCPI 驱动使用。

需要注意的是，在具体产品形态或应用场景中，可能不使能某些 core 或 cluster。为此，示例代码中引入 4 个内存地址：“pSt0”、“pSt1”、“pSt2”、“pSt3”，地址内的数据由 UEFI 填写，表征对应 core 的使能状态。如果 ACPI 描述需要支持上述功能，则 UEFI 必须正确填写这 4 个地址内的值。

```
Scope(_SB)
{
    Device (CLU0) {                    // Cluster0 state
      Name(_HID, "ACPI0010")
      Name(_UID, 1)
      Method (_STA, 0, NotSerialized) {
          if(OR(\pSt0,\pSt1)){
          Return(0x0F)
        }
        else{
          Return(0x00)
        }
      }
      Device(CPU0) {
        Name(_HID, "ACPI0007")
        Name(_UID, 0)
        Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
         Package () {
                Package () {"clock-name","c0"},
                Package () {"clock-domain",0},
         }
        })
        Method (_STA, 0, NotSerialized) {
       if(Lequal(\pSt0,1)){
            Return(0x0F)
       }
       else{
            Return(0x00)
```

```
        }
        }
}

Device(CPU1) {
        Name(_HID, "ACPI0007")
        Name(_UID, 1)
        Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
         Package () {
                Package () {"clock-name","c0"},
                Package () {"clock-domain",0},
         }
        })
        Method (_STA, 0, NotSerialized) {
      if(Lequal(\pSt1,1)){
           Return(0x0F)
      }
      else{
            Return(0x00)
      }
        }
    }
  }

  Device (CLU1) {              // Cluster1 state
     Name(_HID, "ACPI0010")
     Name(_UID, 2)
     Method (_STA, 0, NotSerialized) {
      if(OR(\pSt2,\pSt3)){
           Return(0x0F)
      }
      else{
            Return(0x00)
      }
     }
     Device(CPU2) {
        Name(_HID, "ACPI0007")
        Name(_UID, 2)
        Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
         Package () {
                Package () {"clock-name","c1"},
                Package () {"clock-domain",1},
```

```
        }
        })
        Method (_STA, 0, NotSerialized) {
      if(Lequal(\pSt2,1)){
            Return(0x0F)
      }
      else{
            Return(0x00)
      }
        }
    }

    Device(CPU3) {
        Name(_HID, "ACPI0007")
        Name(_UID, 3)
         Name (_DSD, Package () {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
         Package () {
                Package () {"clock-name","c1"},
                Package () {"clock-domain",1},
        }
        })
        Method (_STA, 0, NotSerialized) {
      if(Lequal(\pSt3,1)){
            Return(0x0F)
      }
      else{
            Return(0x00)
      }
        }
    }
  }
}
```

附录 C PCI Express 的 ACPI 描述

飞腾处理器的 PCIe 模块可能配置为多个 PCIe 控制器。PCIe 的 ACPI 描述示例如下，以 6 个 PCIe 控制器为例，其中 PCI 中断路由表中描述了 6 个 PCIe 控制器的路由配置，_CRS 方法中描述了 PCIe 的 IO、MEM32、MEM64 空间等。

```
#define LNK_DEVICE(Unique_Id, Link_Name, irq)                    \
  Device(Link_Name) {                                  \
      Name(_HID, EISAID("PNP0C0F"))                    \
      Name(_UID, Unique_Id)                            \
      Name(_PRS, ResourceTemplate() {                  \
          Interrupt(ResourceProducer, Level, ActiveHigh, Exclusive，0, "\\_SB.IXIU") { irq }    \
      })                                               \
      Method (_CRS, 0) { Return (_PRS) }               \
      Method (_SRS, 1) { }                             \
      Method (_DIS) { }                                \
  }

#define PRT_ENTRY(Address, Pin, Link)                          \
        Package                                                                                    (4)
 {                                                                                                 \
          Address,   /* uses the same format as _ADR */                                         \
          Pin,       /* The PCI pin number of the device (0-INTA, 1-INTB, 2-INTC, 3-INTD). */   \
          Link,      /* Interrupt allocated via Link device. */                                  \
          Zero       /* global system interrupt number (no used) */              \
  }

#define ROOT_PRT_ENTRY(Dev, Pin, Link)     PRT_ENTRY(Dev * 0x10000 + 0xFFFF, Pin, Link)

DefinitionBlock("SsdtPci.aml",          "SSDT",          1,          "PHYTIUM",          "PHYTIUM          ",
EFI_ACPI_ARM_OEM_REVISION) {
    Scope(_SB) {

    External (\_SB.DBGC, MethodObj)
    //
    // PCI Root Complex
    //
     Device (IXIU) {
        Name (_HID, "PHYT0013")
        Name (_UID, 0)
        Name(_CRS, ResourceTemplate(){
          Memory32Fixed(ReadWrite, 0x29000000, 0x60000)
```

```
            Memory32Fixed(ReadWrite, 0x29100000, 0x2000)
        })

        Name (_DSD, Package () {
            ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
            Package () {
                Package () {"intx-spi-base", 60},
            }
        })
}
  LNK_DEVICE(1, LNKA, 60)
  LNK_DEVICE(2, LNKB, 61)
  LNK_DEVICE(3, LNKC, 62)
  LNK_DEVICE(4, LNKD, 63)

  // reserve ECAM memory range
  Device(RES0)
  {
      Name(_HID, EISAID("PNP0C02"))
      Name(_UID, 0)
      Name(_CRS, ResourceTemplate() {

       QWordMemory (ResourceConsumer, PosDecode, MinFixed, MaxFixed, Cacheable, ReadWrite,
            0x0, // Granularity
            0x40000000, // Range Minimum
            0x4FFFFFFF, // Range Maximum
            0, // Translation Offset
            0x10000000, // Length
            ,,)
      })
  }

  Device(PCI0)
  {
      Name(_HID, EISAID("PNP0A08")) // PCI Express Root Bridge
      Name(_CID, EISAID("PNP0A03")) // Compatible PCI Root Bridge
      Name(_SEG, Zero) // PCI Segment Group number
      Name(_BBN, 0)       // PCI Base Bus Number
      Name(_CCA, 1)       // Initially mark the PCI coherent (for JunoR1)

      // Root Complex
      Device (RP0) {
          Name(_ADR, 0x00000000)       // Dev 0, Func 0
      }
```

```
// PCI Routing Table
Name(_PRT, Package() {
    ROOT_PRT_ENTRY(0, 0, LNKA),     // INTA
    ROOT_PRT_ENTRY(0, 1, LNKB),     // INTB
    ROOT_PRT_ENTRY(0, 2, LNKC),     // INTC
    ROOT_PRT_ENTRY(0, 3, LNKD),     // INTD

    ROOT_PRT_ENTRY(1, 0, LNKA),     // INTA
    ROOT_PRT_ENTRY(1, 1, LNKB),     // INTB
    ROOT_PRT_ENTRY(1, 2, LNKC),     // INTC
    ROOT_PRT_ENTRY(1, 3, LNKD),     // INTD

    ROOT_PRT_ENTRY(2, 0, LNKA),     // INTA
    ROOT_PRT_ENTRY(2, 1, LNKB),     // INTB
    ROOT_PRT_ENTRY(2, 2, LNKC),     // INTC
    ROOT_PRT_ENTRY(2, 3, LNKD),     // INTD

    ROOT_PRT_ENTRY(3, 0, LNKA),     // INTA
    ROOT_PRT_ENTRY(3, 1, LNKB),     // INTB
    ROOT_PRT_ENTRY(3, 2, LNKC),     // INTC
    ROOT_PRT_ENTRY(3, 3, LNKD),     // INTD

    ROOT_PRT_ENTRY(4, 0, LNKA),     // INTA
    ROOT_PRT_ENTRY(4, 1, LNKB),     // INTB
    ROOT_PRT_ENTRY(4, 2, LNKC),     // INTC
    ROOT_PRT_ENTRY(4, 3, LNKD),     // INTD

   ROOT_PRT_ENTRY(5, 0, LNKA),     // INTA
    ROOT_PRT_ENTRY(5, 1, LNKB),     // INTB
    ROOT_PRT_ENTRY(5, 2, LNKC),     // INTC
    ROOT_PRT_ENTRY(5, 3, LNKD),     // INTD


})

// Root complex resources
Method (_CRS, 0, Serialized) {
    Name (RBUF, ResourceTemplate () {
        WordBusNumber ( // Bus numbers assigned to this root
            ResourceProducer,
            MinFixed, MaxFixed, PosDecode,
            0,      // AddressGranularity
            0,      // AddressMinimum - Minimum Bus Number
            255, // AddressMaximum - Maximum Bus Number
```

```
            0,      // AddressTranslation - Set to 0
            256     // RangeLength - Number of Busses
        )

        DWordMemory ( // 32-bit BAR Windows
            ResourceProducer, PosDecode,
            MinFixed, MaxFixed,
            Cacheable, ReadWrite,
            0x00000000,                 // Granularity
            0x58000000,                 // Min Base Address
            0x7FFFFFFF,                  // Max Base Address
            0x00000000,                 // Translate
            0x28000000                   // Length
        )

        QWordMemory ( // 64-bit BAR Windows
            ResourceProducer, PosDecode,
            MinFixed, MaxFixed,
            Cacheable, ReadWrite,
            0x00000000,                 // Granularity
            0x1000000000,                // Min Base Address
            0x1FFFFFFFFF,                 // Max Base Address
            0x0000000000,                // Translate
            0x1000000000                  // Length
        )

        DWordIo ( // IO window
            ResourceProducer,
            MinFixed,
            MaxFixed,
            PosDecode,
            EntireRange,
            0x00000000,                 // Granularity
            0x00000000,                 // Min Base Address
            0x00efffff,                // Max Base Address
            0x50000000,                 // Translate
            0x00f00000,                  // Length
            ,,,TypeTranslation
        )
    }) // Name(RBUF)

    Return (RBUF)
} // Method(_CRS)
```

```
//
// OS Control Handoff
//
Name(SUPP, Zero) // PCI _OSC Support Field value
Name(CTRL, Zero) // PCI _OSC Control Field value

Method(_OSC,4) {
  // Check for proper UUID
  If(LEqual(Arg0,ToUUID("33DB4D5B-1FF7-401C-9657-7441C03DD766"))) {
    // Create DWord-adressable fields from the Capabilities Buffer
    CreateDWordField(Arg3,0,CDW1)
    CreateDWordField(Arg3,4,CDW2)
    CreateDWordField(Arg3,8,CDW3)

    // Save Capabilities DWord2 & 3
    Store(CDW2,SUPP)
    Store(CDW3,CTRL)
    If(LNotEqual(And(SUPP, 0x16), 0x16)) {
      And(CTRL,0x1E,CTRL) // Mask bit 0 (and undefined bits)
    }

    And(CTRL,0x10,CTRL)

    If(LNotEqual(Arg1,One)) {    // Unknown revision
      Or(CDW1,0x08,CDW1)
    }

    If(LNotEqual(CDW3,CTRL)) {    // Capabilities bits were masked
      Or(CDW1,0x10,CDW1)
    }
    // Update DWORD3 in the buffer
    Store(CTRL,CDW3)
    //DBGC(1, "_OSC PCI",0)
    //DBGC(2, "CDW1:", CDW1)
    //DBGC(2, "CDW2:", CDW2)
    //DBGC(2, "CDW3:", CDW3)
    Return(Arg3)
  } Else {
    Or(CDW1,4,CDW1) // Unrecognized UUID
    Return(Arg3)
  }
} // End _OSC
} // PCI0
```

```
    }
}
```

附录 D EC 的 ACPI 描述

基于飞腾处理器的整机系统，可能采用嵌入式控制器（Embedded Controller，EC），实现系统电源控制，以及键盘、电池等外设的管理。EC 连接到处理器的 LPC 总线上，EC 下连接了键盘、触摸板、电池、温度传感器等设备。

系统软件，比如固件或操作系统，通过 EC 标准接口[6]访问 EC 下的资源。针对不同的具体实现，固件可以通过 ACPI 表对 EC 接口进行定制化描述，比如，可以通过“ec_battery_remain_percent”描述电池的剩余电量百分比的访问接口，通过“ec_cpu_temp”描述 CPU 温度的访问接口。

```
Scope(_SB)
{
  Device (LPC1) {
    Name (_HID, "PHYT000B")       /* LPC */
    Name (_CID, "LPC0001")        /* compatible with v1.0 */

    Name (_UID, Zero)
    Name (_CRS, ResourceTemplate () {
      Memory32Fixed (ReadWrite, 0x20000000, 0x08000000,)
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive, ,, ) { 0x25 }
    })

 Device(KBC) {      /* Keyboard Controller */
      Name(_HID, "KBCI8042")
      Name(_UID, 0)
      Name(_DSD, Package() {
        ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
        Package(0x02){
          Package(0x02){"i8042_command_reg",0x64},
          Package(0x02){"i8042_data_reg",0x60},
      }
       })
      }

    Device (BAT1) {
      Name (_HID, "PHYT000C")       /* Battery */
      Name (_CID, "FTEC0002")       /* compatible with v1.0 */

      Name (_UID, Zero)
      Name (_CRS, ResourceTemplate ()   {
      })
```

```
        Name(_DSD, Package()
        {
            ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),Package(0x02){
                    Package(0x02){"ec_battery_remain_percent",0x21},  //电池剩余百分比电量
            }
        }
    }

    Device (SEN3) {
        Name (_HID, "PHYT000D")         /* Sensor */
        Name (_CID, "FTEC0003")         /* compatible with v1.0 */
        Name (_UID, Zero)
        Name (_CRS, ResourceTemplate ()    {
        })
         Name(_DSD, Package()
         {
              ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),Package(0x02){
                  Package(0x02){"ec_cpu_temp",0x98},   //EC CPU 温度
          }

    }

    Device(EC0) {
        Name (_HID, "PHYT000B")         /* EC */
        Name (_CID, "FTEC0001")         /* compatible with v1.0 */
        Name(_UID, Zero)
        Name(_CRS, ResourceTemplate() {
          GpioInt(Edge, ActiveLow, ExclusiveAndWake, PullUp,,"\\_SB.GPI0") {7}
        })

        Name(_DSD, Package() {
          ToUUID("daffd814-6eba-4d8c-8a91-bc9bbf4aa301"),
          Package() {
            Package(){"gpio-sci", Package() {^EC0, 0, 0, 1}},
            }
        })
    }
  }
}
```