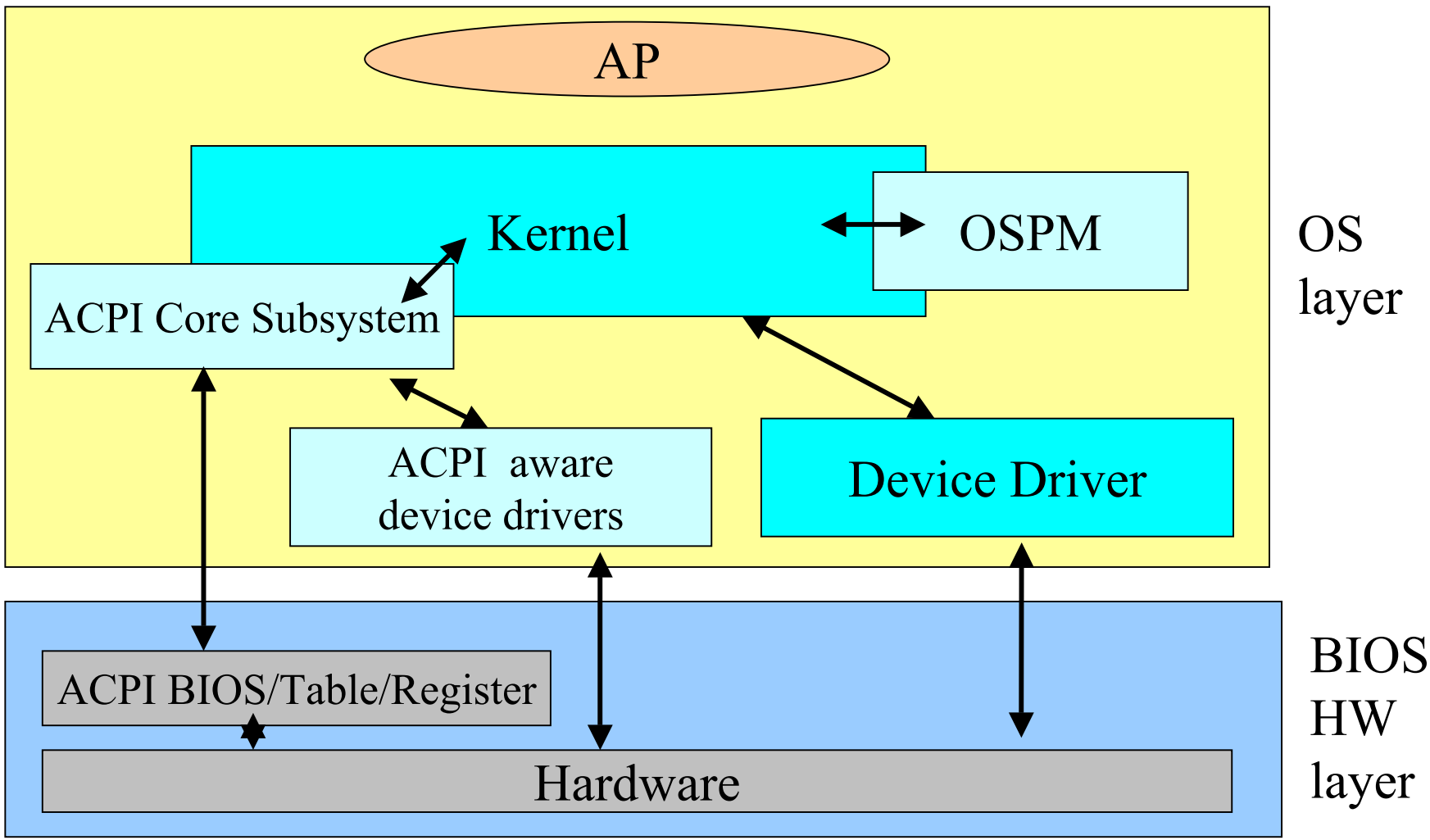
LinuxにおけるACPI構造の解説

NECシステムテクノロジー株式会社
夜久　健一

ACPIとは

- ACPIの特徴
  - 電源管理を行う
  - ACPIは、OSがパワーマネージメント
  - 本体装置、ソフトウェアの連携

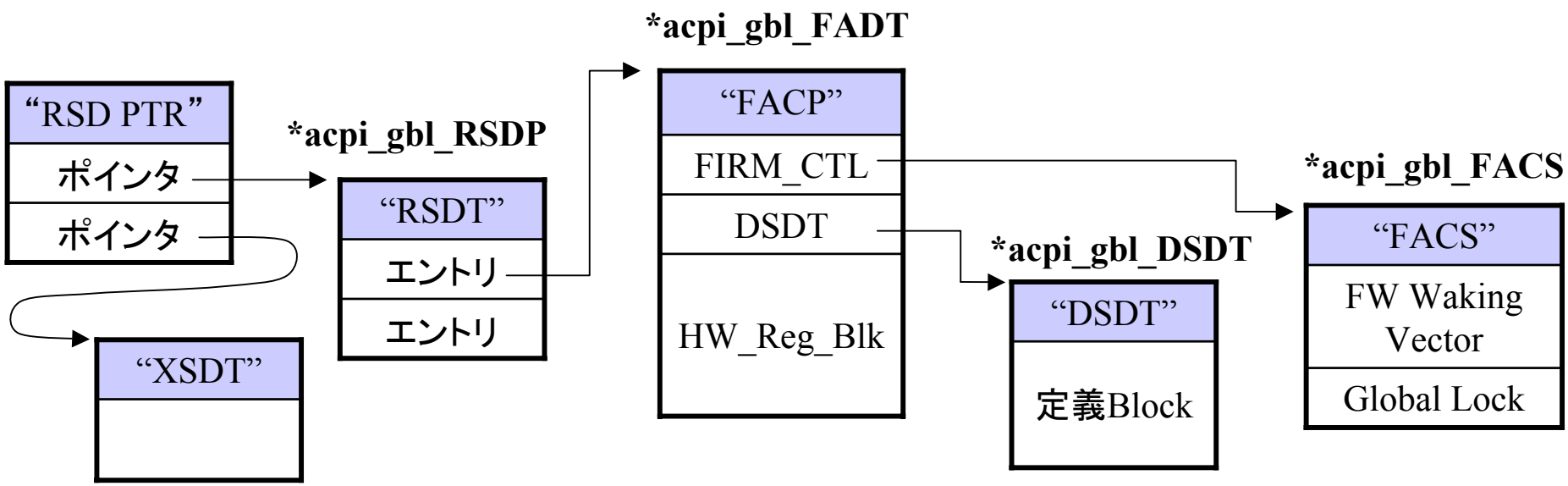
ACPI構造


AP
Kernel
OSPM
ACPI Core Subsystem
ACPI aware
device drivers
Device Driver
OS
layer
ACPI BIOS/Table/Register
Hardware
BIOS
HW
layer

BIOS Memory MAP

```
>> cat dmesg | less
 BIOS-provided physical RAM map:
   BIOS-e820:  0000000000000000 - 000000000009f800 (usable)
   BIOS-e820:  000000000009f800 - 00000000000a0000 (reserved)
   BIOS-e820:  00000000000e4400 – 0000000000100000 (reserved)
   BIOS-e820:  0000000000100000 – 0000000017ff0000 (usable)
   BIOS-e820:  0000000017ff0000 – 0000000017ffffc0 (ACPI data)
   BIOS-e820:  00000000fffe0000 – 0000000100000000 (reserved)
```

ACPI Table管理



RSD PTR … Root System Description Pointer

RSDT …….. Root System Description Table

XSDT …….. Extended System Description Table

FACP …….. Fixed ACPI Description Table

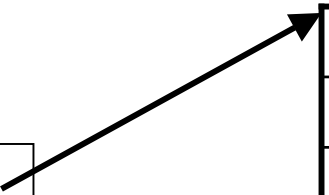
DSDT …….. Differentiated System Description Table

FACS …….. Firmware ACPI Control Structure

acpi_gbl_acpi_tables[]

acpi_table_desc
acpi_gbl_acpi_tables[]

[0]	ACPI_TABLE_RSDP
[1]	ACPI_TABLE_DSDT
[2]	ACPI_TABLE_FADT
[3]	ACPI_TABLE_FACS
[4]	ACPI_TABLE_PSDT
[5]	ACPI_TABLE_SSDT
[6]	ACPI_TABLE_XSDT

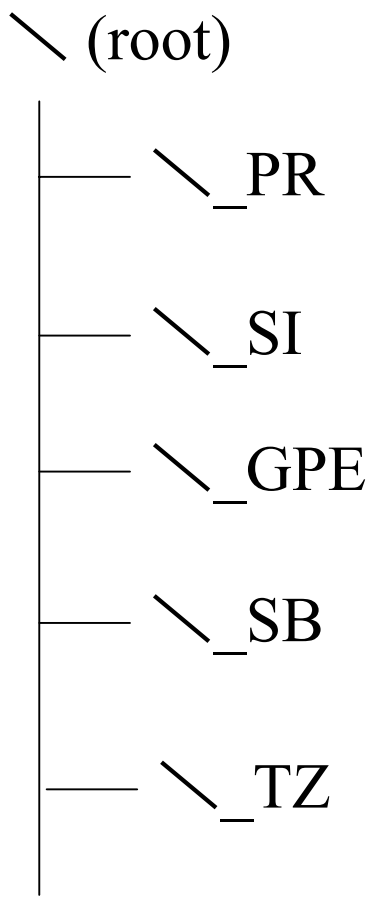


acpi_table_desc構造体

struct acpi_table_desc	*prev
struct acpi_table_desc	*next
struct acpi_table_desc	*installed_desc
acpi_table_header	*pointer
void	*base_pointer
u8	*aml_start
u64	physical_address
u32	aml_length
ACPI_SIZE	length
u32	count
acpi_owner_id	table_id
u8	type
u8	allocation
u8	loaded_into_namespace

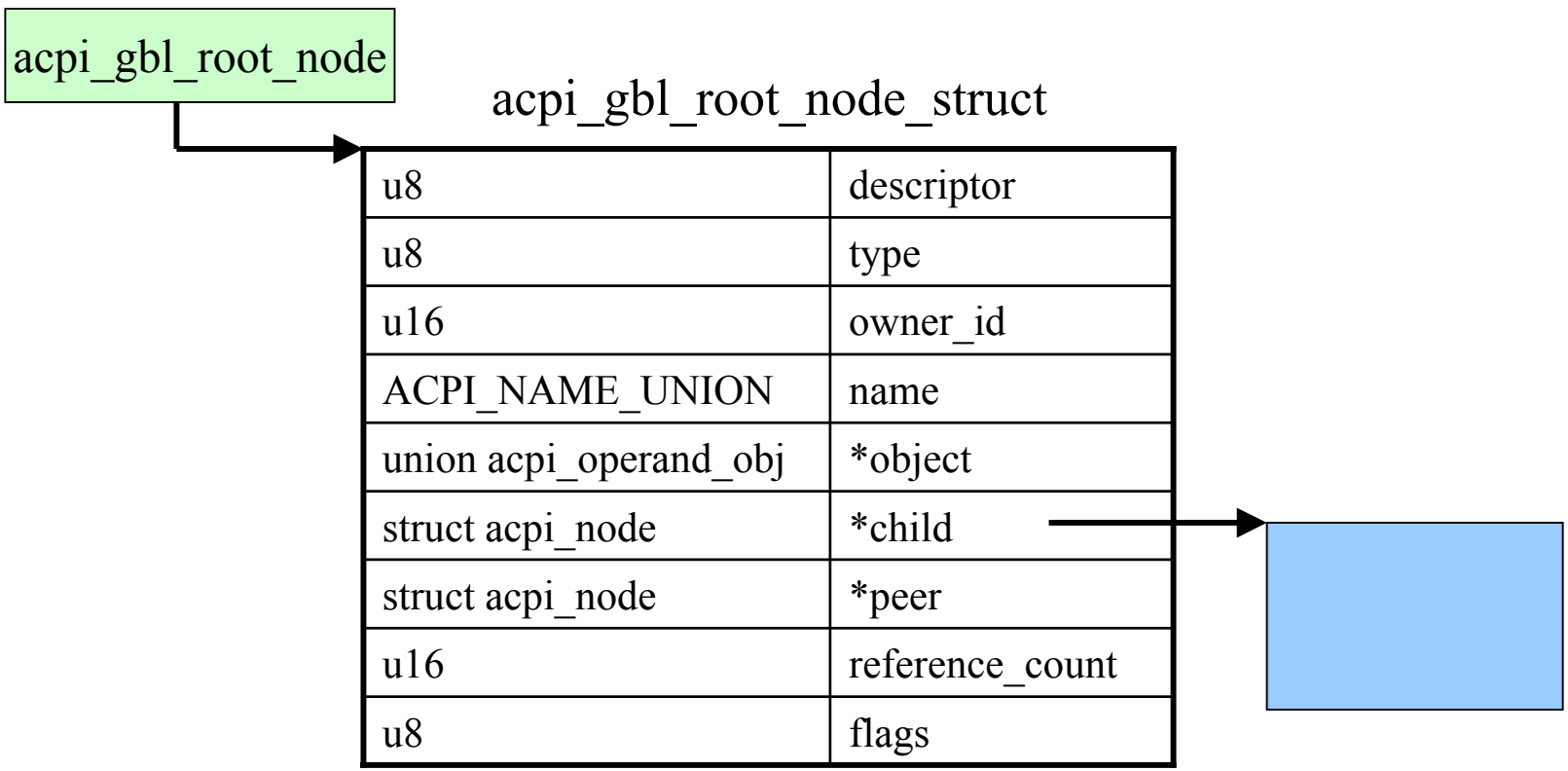
Namespace

- システムは定義ブロックを一つの階層構造を持ったNamespaceを管理する。
- 詳細なHWの情報(データ/AML(ASL) /その他オブジェクト)を含んだ定義ブロックは、後にNameSpace内に配置される。

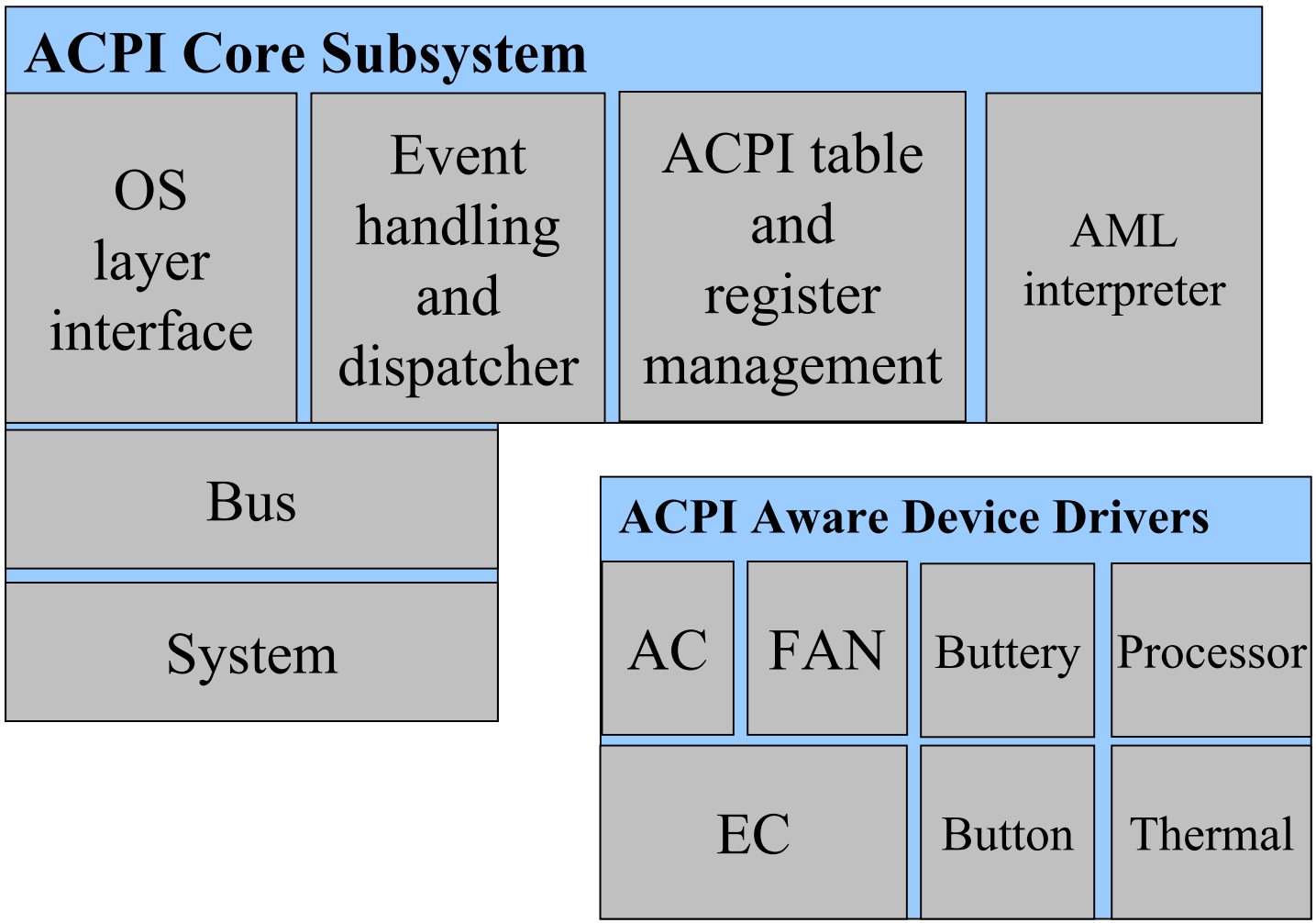


acpi_gbl_root_node

Namespaceはacpi_gbl_root_nodeで管理



ACPIドライバ構造


ACPI Core Subsystem
OS layer interface
Event handling and dispatcher
ACPI table and register management
AML interpreter
Bus
System
ACPI Aware Device Drivers
AC
FAN
Buttery
Processor
EC
Button
Thermal

Synthesize
NECシステムテクノロジー

Power Management

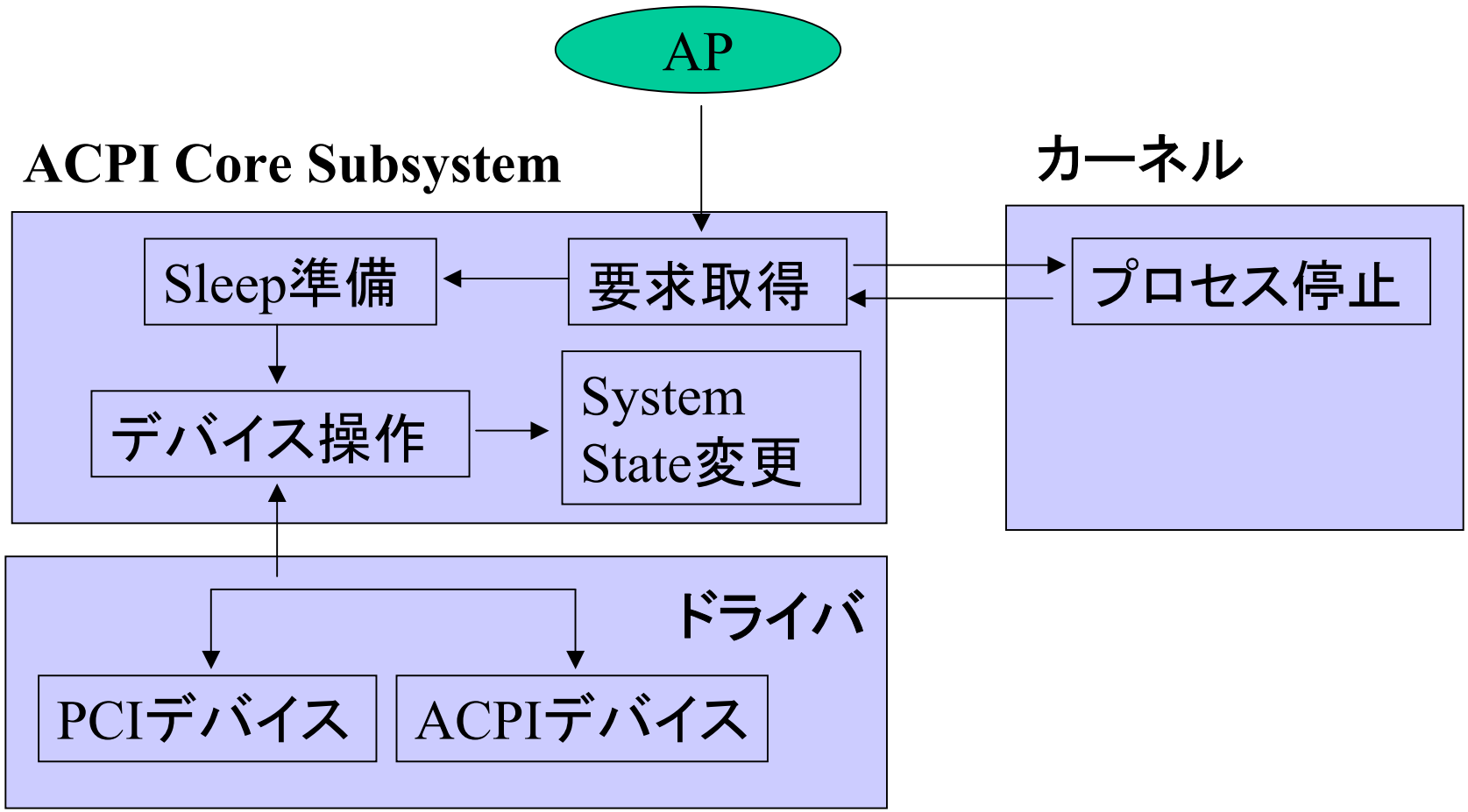
全体/スリープステート

ステート		動作状態
G3		ハードウェアオフ
G2/S5		ソフトオフ
G1	S1	スリープ
	S2	サスペンド
	S3	サスペンド。STR
	S4	STD。ハイバネーション（BIOS,OS)
G0/S0		動作中

システムのパワーマネージメント

- ACPI Core Subsystem内のモジュール system driverで制御する
- System Driverの機能
  - SystemをSleepモードにする
  - /procインタフェースの提供
  - Busドライバとのインタフェース

動作概要例


AP
ACPI Core Subsystem
カーネル
Sleep準備
要求取得
プロセス停止
デバイス操作
System
State変更
ドライバ
PCIデバイス
ACPIデバイス

Sleep要求処理

- /proc/acpi//sleepにWriteする
  - acpi_system_write_sleep()
    - S4が指定された場合は、softwake_suspend()を実行
    - それ以外の場合acpi_suspend()を呼び出して処理する
      - 動作中のプロセスをFreeze状態にする(freeze_processes())
      - S2 or S3 の場合、wake upベクタを設定する。
      - _PTS(prepare to sleep), _GTS(go to sleep)メソッドを実行(acpi_enter_sleep_state_prep())
      - デバイスの状態をセーブ(acpi_system_save_state(state))
      - 割込みDisable(“cli”)とCPU CACHEをフラッシュする
      - acpi_system_suspend()
        - S1の場合、S1ステートにする。(acpi_enter_sleep_state(ACPI_STATE_S1))
        - S2, S3にする場合、do_suspend_lowlevel(0)を実行する。
          - プロセッサコンテキストをセーブする。save_processor_context()、
          - 復帰アドレスをセーブする。(acpi_save_register_state(&&acpi_sleep_done))
          - S3ステートにする。(acpi_enter_sleep_state(3))

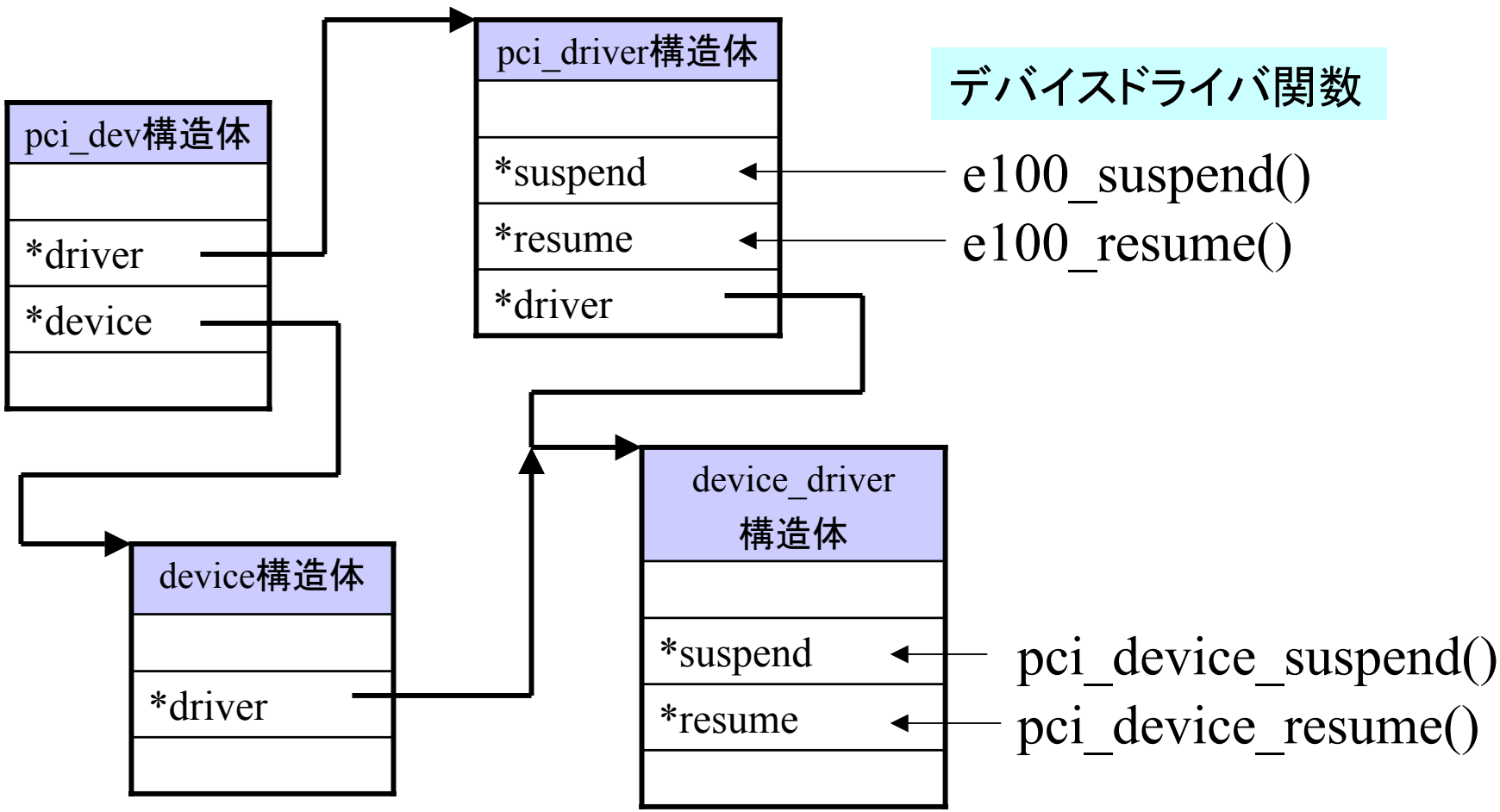
デバイスのSleep処理

- acpi_system_save_state()
  - device_suspend(state, SUSPEND_NOTIFY)
    - dev->driver->suspend(dev,state,SUSPEND_NOTIFY)を実行し、デバイスに通知
  - State < ACPI_STATE_S5 の場合
    - device_suspend(state, SUSPEND_SAVE_STATE)
    - ACPI_FLUSH_CPU_CACHE()
    - state > ACPI_STATE_S1 の場合
      - acip_save_state_mem()　wakeupルーチンのCopy
      - device_resume(RESUME_RESTORE_STATE)
  - ACPI_DISABLE_IRQS()
  - device_suspend(state, SUSPEND_POWER_DOWN)
  - エラー等が発生した場合、acpi_system_restore_state(state)を実行

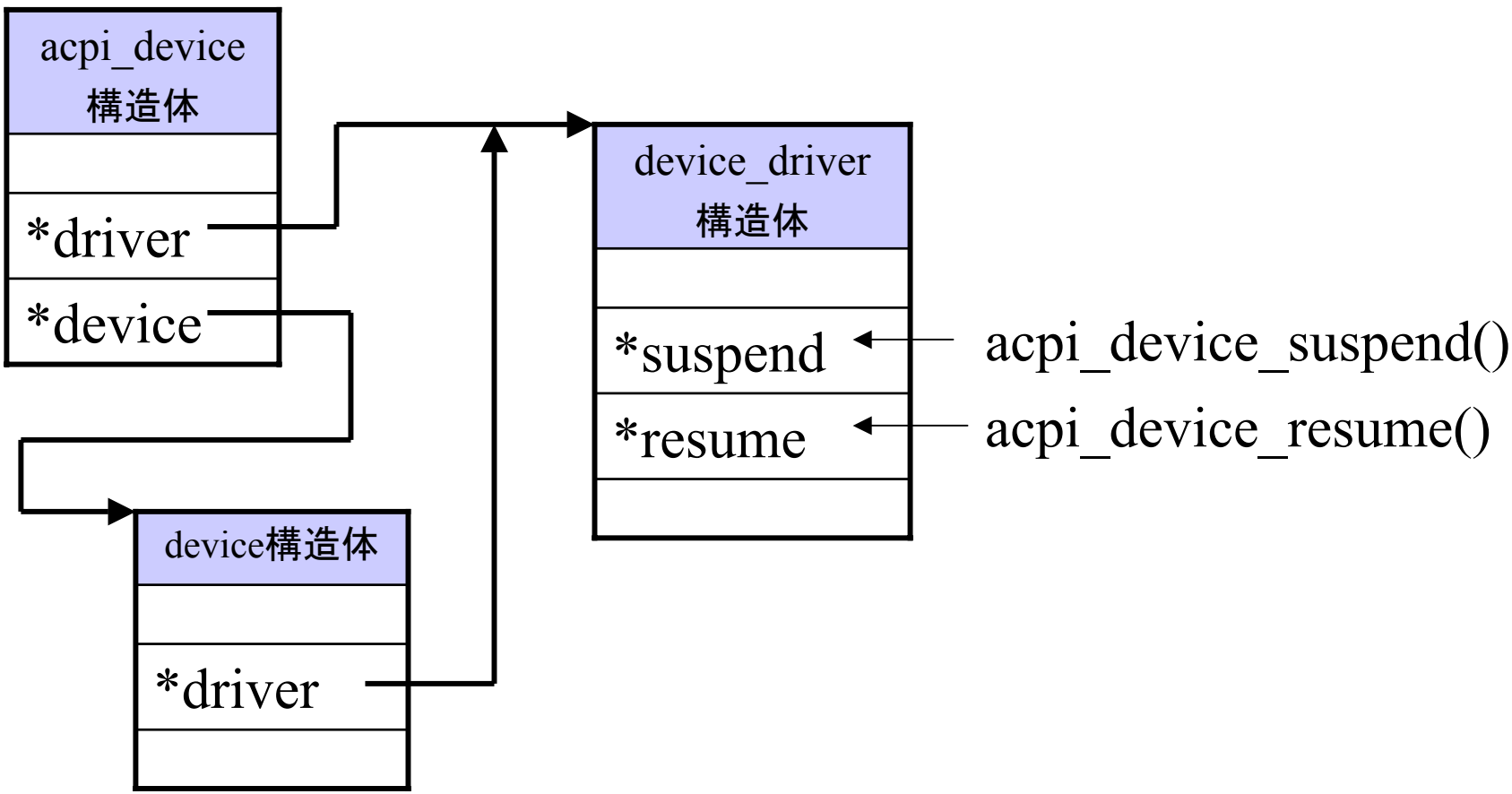
ACPIにおけるデバイス管理

- デバイスは、各バス毎に管理される。PCIデバイスは、PCIバス毎に、ACPIデバイス(例えば、PowerボタンやSleepボタンなど)は、ACPIバス配下の管理となる。
- デバイスは、Genericデバイス構造で管理される。
- ACPIでのデバイスサーチは、global_device_listを使用して行われる。

PCIデバイスの構造

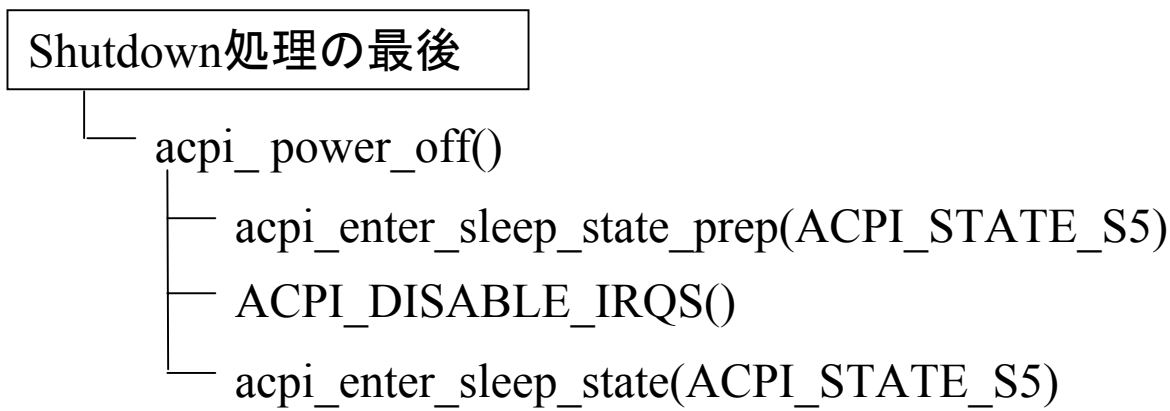

pci_dev構造体
*driver
*device
pci_driver構造体
*suspend
*resume
*driver
デバイスドライバ関数
e100_suspend()
e100_resume()
device構造体
*driver
device_driver
構造体
*suspend
*resume
pci_device_suspend()
pci_device_resume()

ACPIデバイスの構造



Shutdown後電源OFF

Shutdown処理の最後で、acpi_power_off()関数がCallされる。その後、電源OFF処理を実行するために、システムをS5ステートに移行する。



Wakeup

- S1は、ACPIドライバに復帰する。
- S2,S3からの復帰は、CPUのリセットベクタを実行し復帰する。その後、Firmwareのwakeupベクタを実行し、OSへ復帰する
- OS制御によるS4からの復帰は、システム起動時のカーネル初期化フェーズで、レジューム処理を行い復帰する
- S5の場合は、通常のOS起動と同じ

デバイス/プロセッサステート

デバイス

ステート	動作状態
D3	オフ
D2/D1	省電力（デバイスに依存）
D0	フル

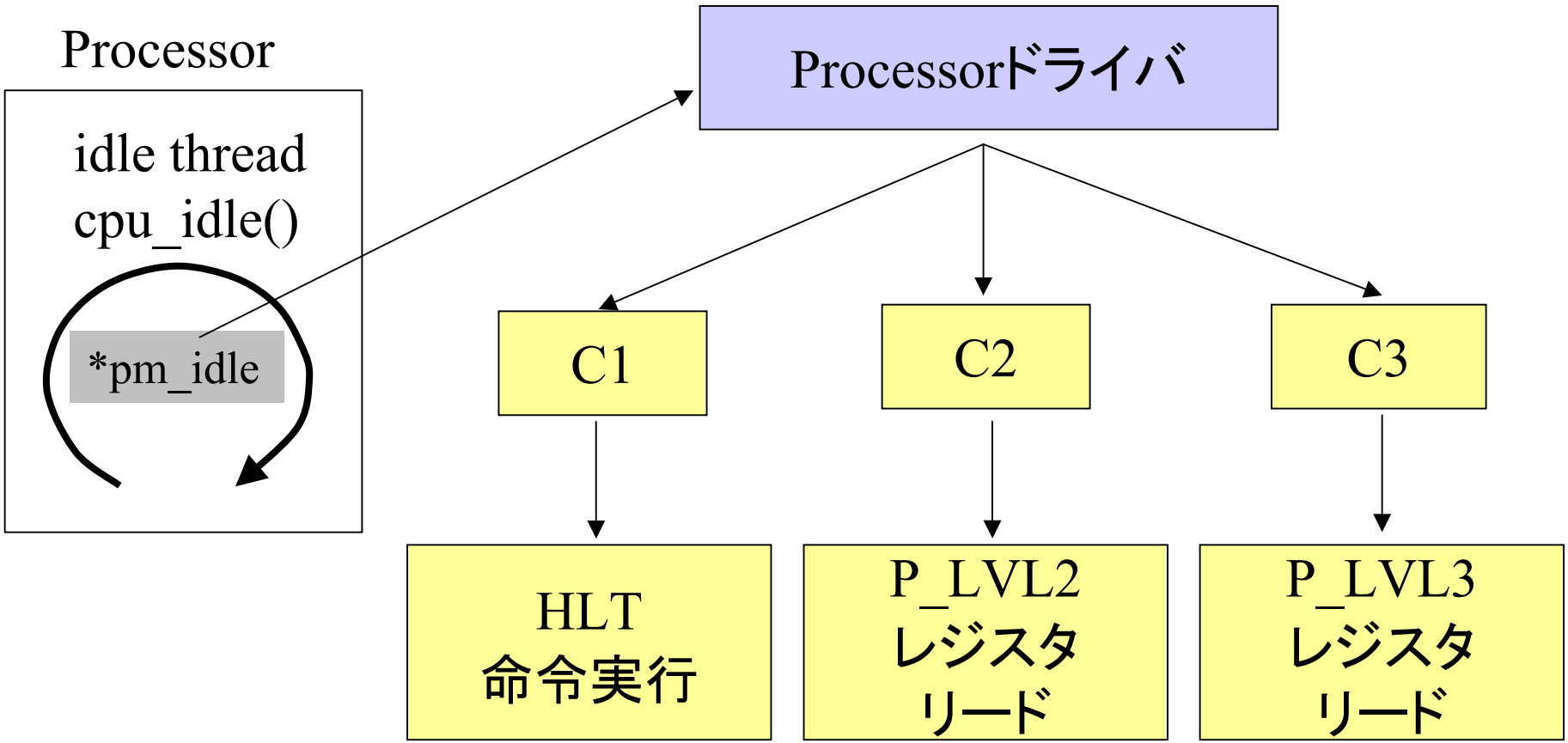
プロセッサ

ステート	動作状態
C3/C2	省電力
C1	HLT命令実行
C0	フル

プロセッサドライバ

- プロセッサのステートを制御する
- 機能
  - Power Management
  - Performance Management
  - Throttling Control
  - Limit Interface
  - /procインタフェース
  - Driverインタフェース

プロセッサコントロール



Synthesize
NECシステムテクノロジー

Event Handling

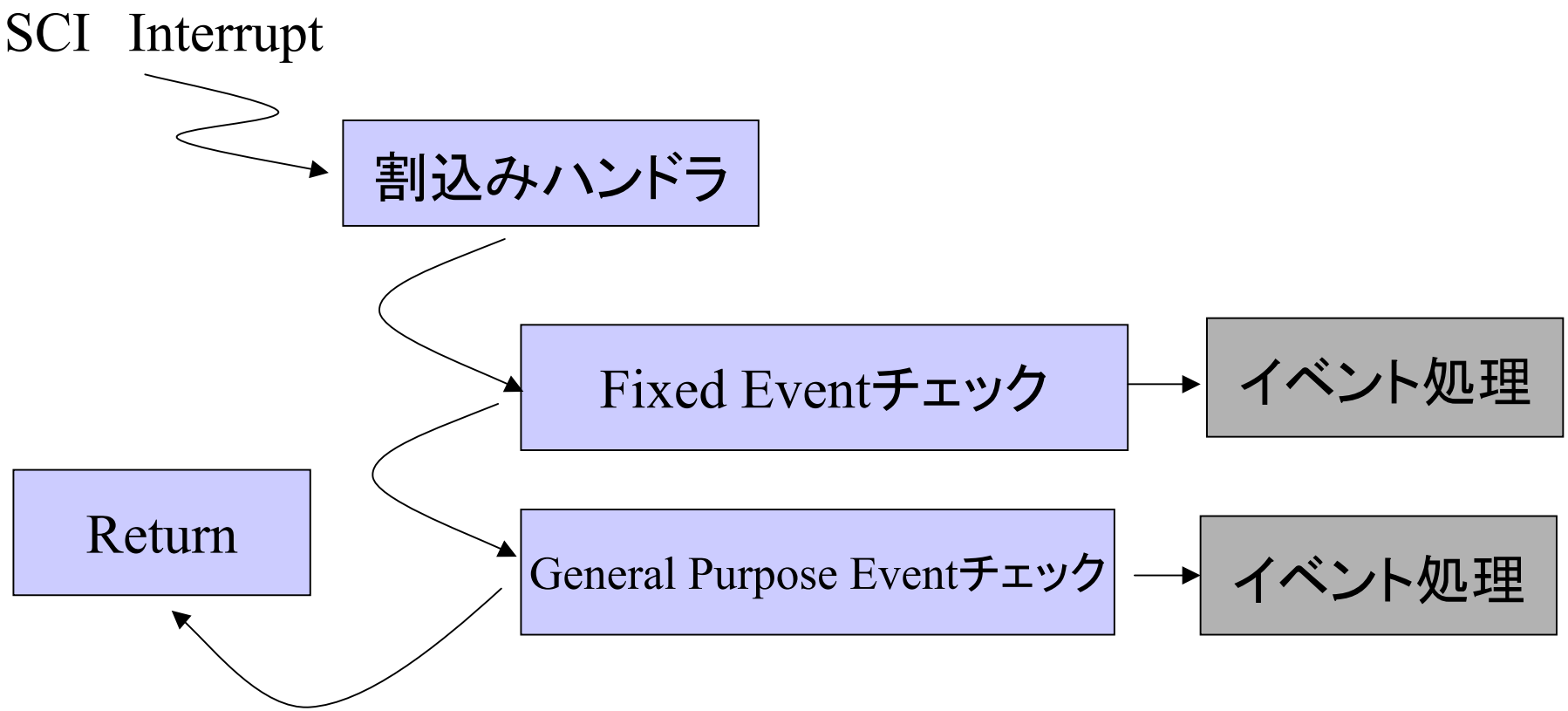
イベント種類

- OS透過イベント
- 割り込みイベント

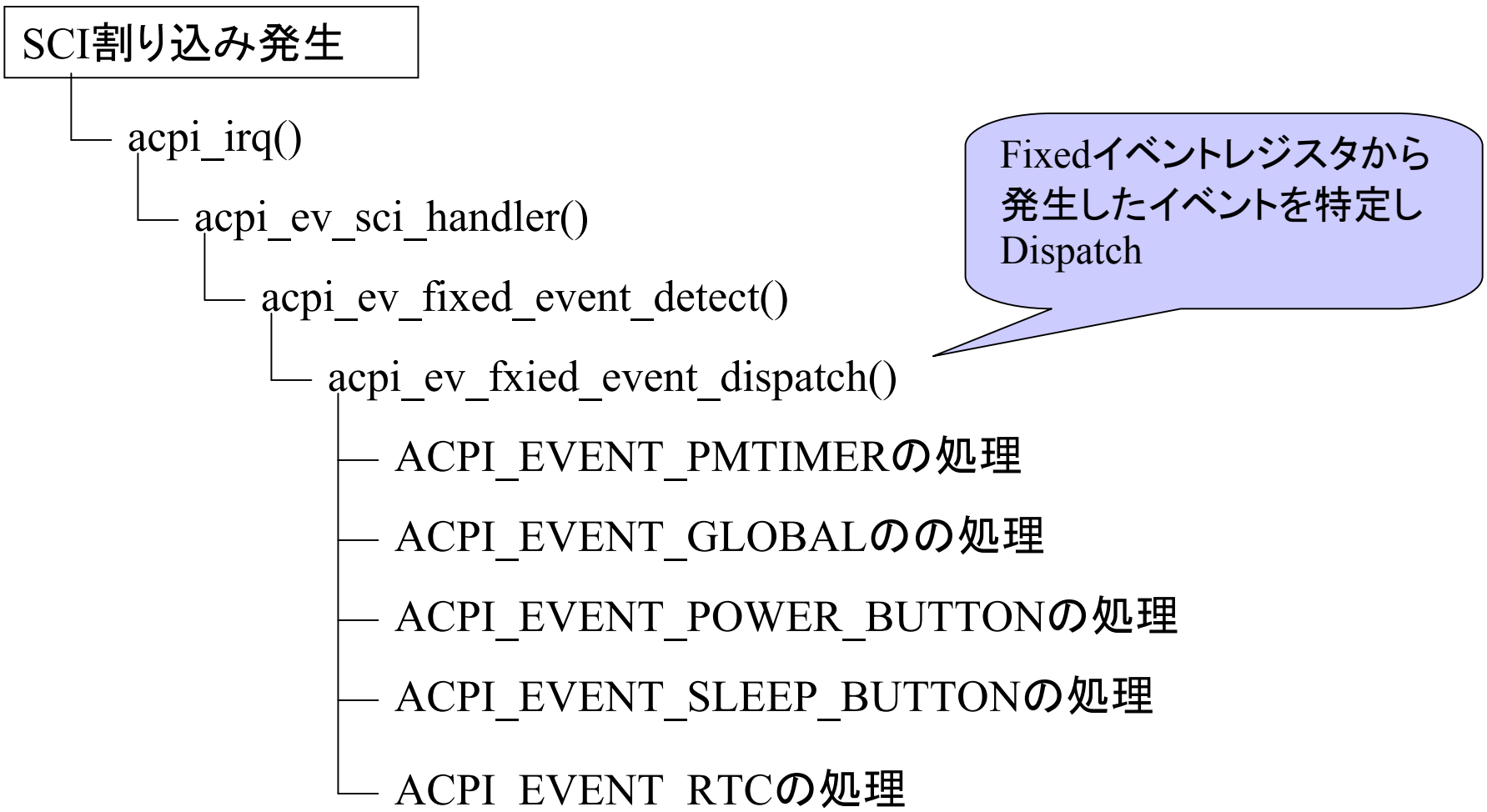
割り込みイベント

- SCI割り込みによって処理される
  - Fixed　ACPI　イベント
    - ACPIドライバが直接イベントレジスタにアクセスしハンドリング
  - General Purpose イベント
    - ACPIドライバがイベントを受け取り、ACPI-awareドライバがハンドリングするか、もしくはAML制御メソッドがイベントをハンドリング

割り込み処理

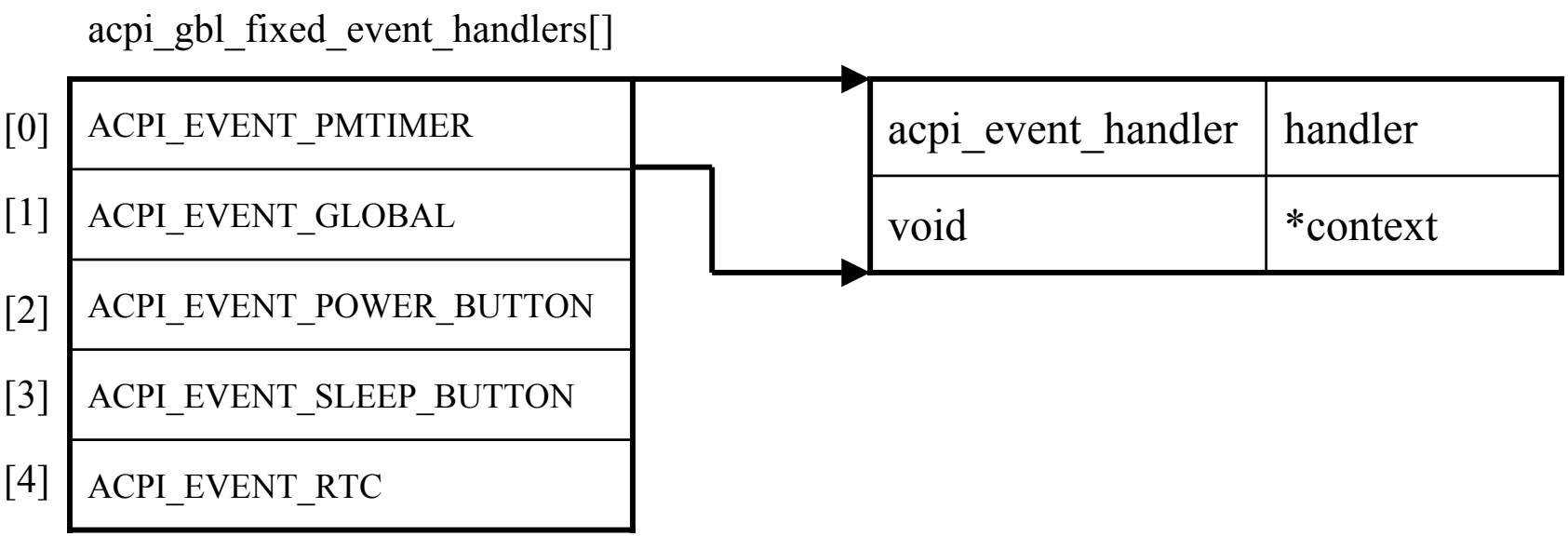

SCI Interrupt
割込みハンドラ
Fixed Eventチェック
イベント処理
Return
General Purpose Eventチェック
イベント処理

Fixedイベント割り込み処理流れ



Fixedイベントハンドラの管理

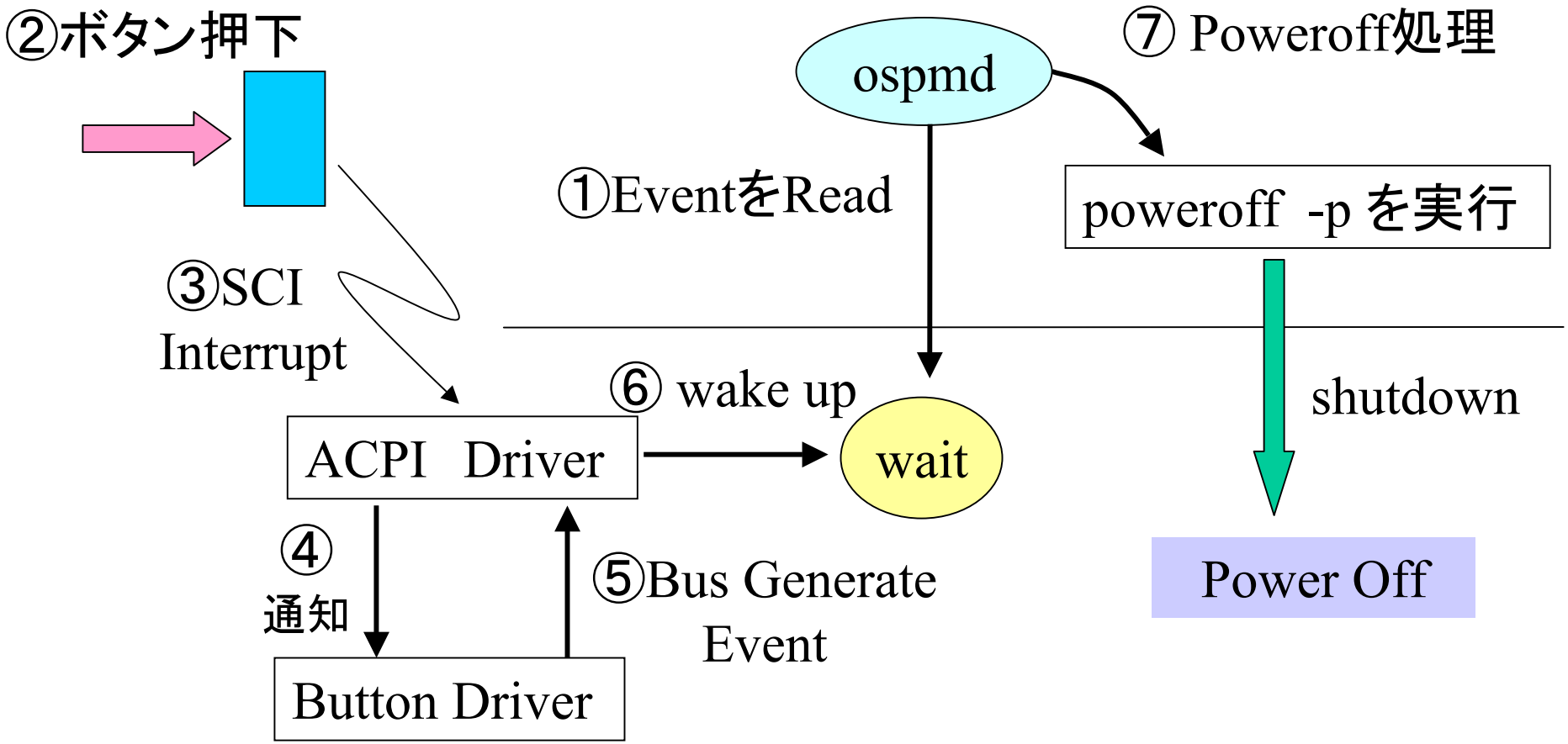
Fixedイベントを処理するハンドラは、
acpi_gbl_fixed_event_handlers[]で管理される。



Fixedイベントハンドラの登録

- acpi_install_fixed_event_handler()関数で登録
- ３つのイベントハンドラが登録される
  - ACPI_EVENT_GLOBAL
    - acpi_ev_global_lock_handler()
  - ACPI_BUTTON_TYPE_POWERF
    - acpi_button_notify_fixed()
  - ACPI_ BUTTON_TYPE_SLEEPF
    - acpi_button_notify_fixed()

電源ボタン押下時の処理



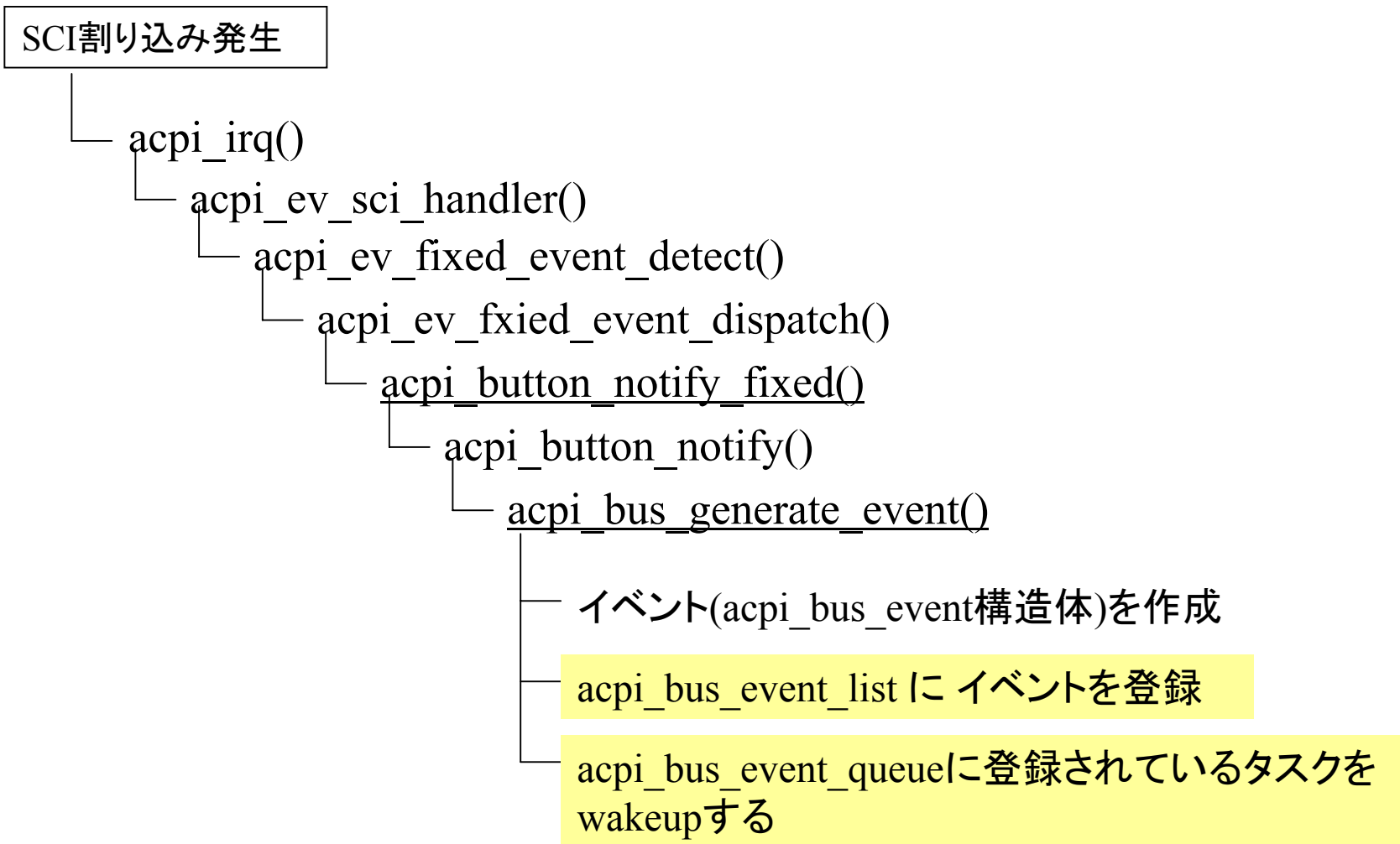
プロセス側から見た処理

/proc/acpi/eventをReadする

- acpi_system_read_event()
  - acpi_bus_receive_event()
    - acpi_bus_event_queue にタスクを登録して寝る
    - wakeupされた場合、バスイベントリスト(acpi_bus_event_list)からイベントを取り出す
  - 取り出したイベントをプロセスに返す

バスイベントはSCI割り込みの延長で登録される。
acpi_bus_generate_event()

Powerボタンイベントの処理



GPE割り込み処理流れ1

SCI割り込み発生

- acpi_irq()
  - acpi_ev_sci_handler()
    - acpi_ev_gpe_detect()

イベントに応じてDispatch。

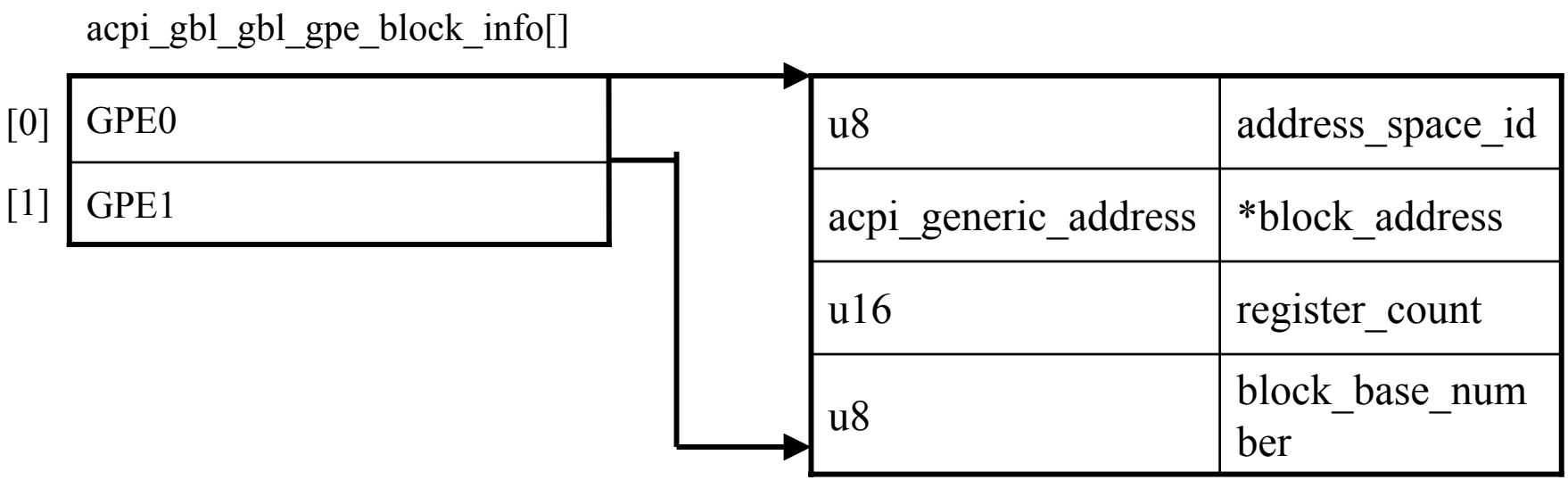
- 8bitのGPEステータスレジスタとEnableレジスタを順番に全て読む
- StatusレジスタがEnableかどうか確認する。 Enableでなければ次のレジスタを読む
- StatusレジスタがEnableのbit（イベント)について、処理を行う。

  acpi_ev_gpe_dispatch()
- returnする

acpi_gbl_gpe_block_info[]

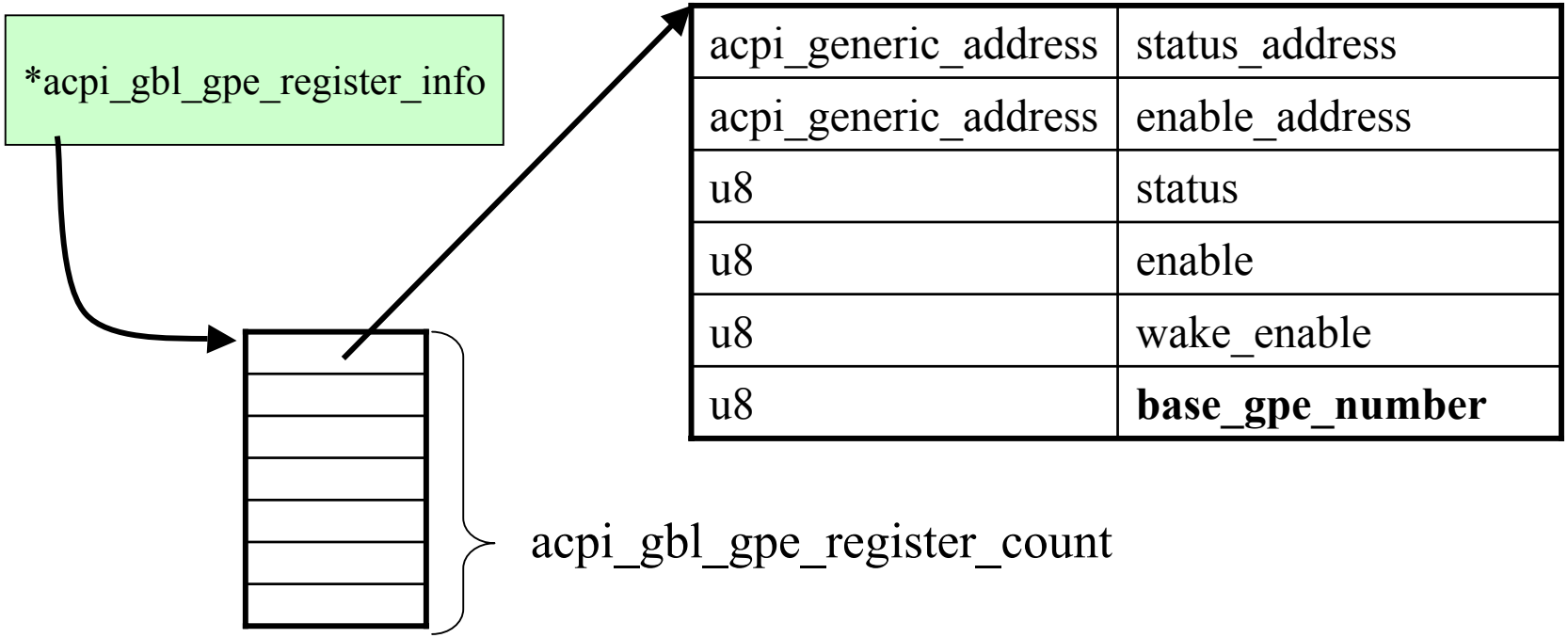
FADTのGPE0ブロックとGPE1ブロックを管理する構造体配列。

それぞれのGeneral Purpose Eventレジスタのアドレス位置、ブロック位置、レジスタ数、ベース番号(GPE0のベース番号は0)を管理する。



acpi_gbl_gpe_register_info

GPEレジスタを管理する。



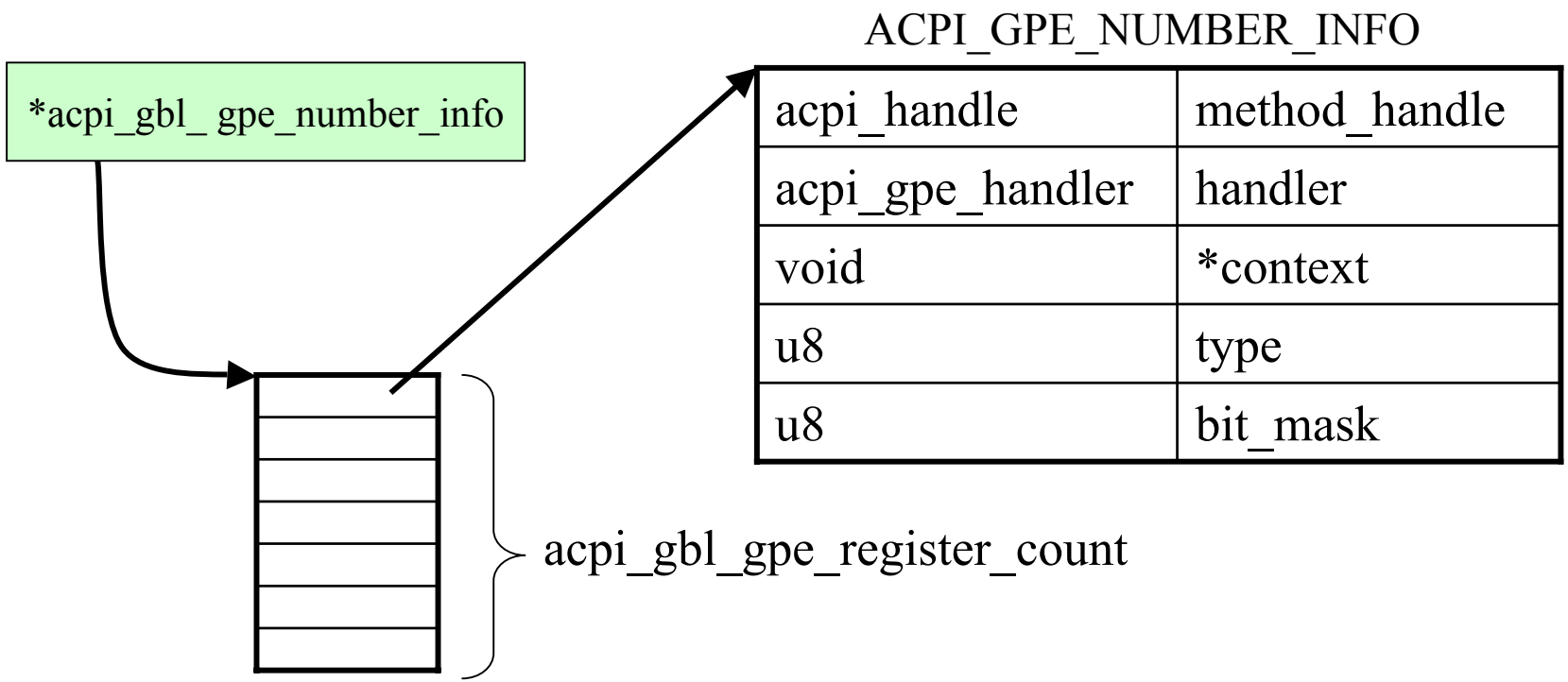
ACPI_GPE_REGISTER_INFO

acpi_generic_address	status_address
acpi_generic_address	enable_address
u8	status
u8	enable
u8	wake_enable
u8	**base_gpe_number**

acpi_gbl_gpe_number_info

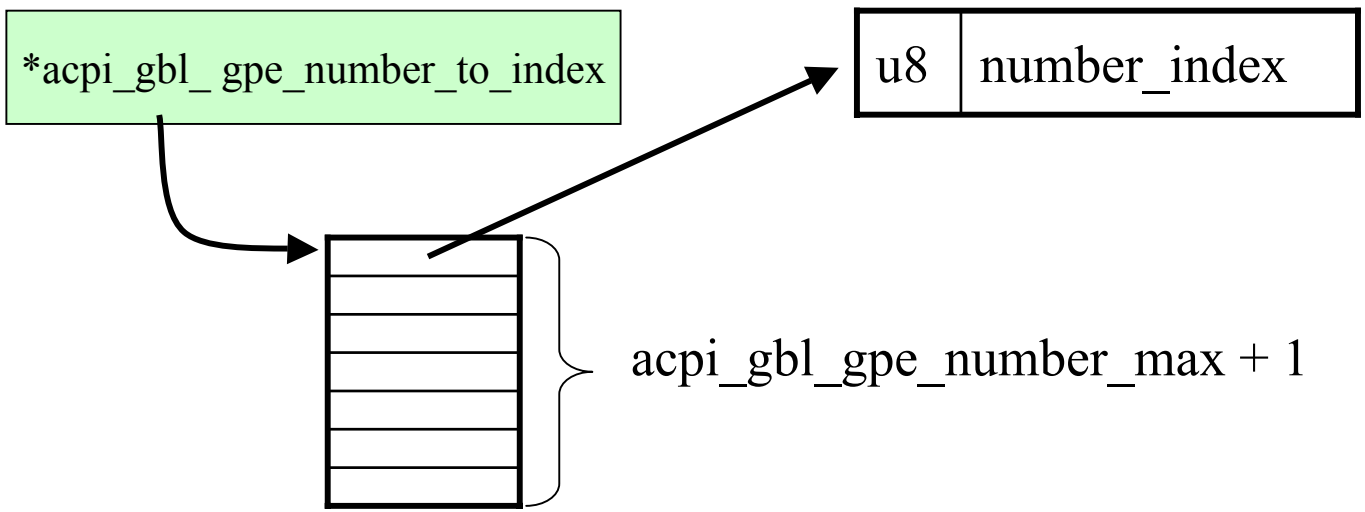
GPE dipatchハンドラブロックを管理。

method_handleは、＼_GPE配下のメソッドオブジェクト

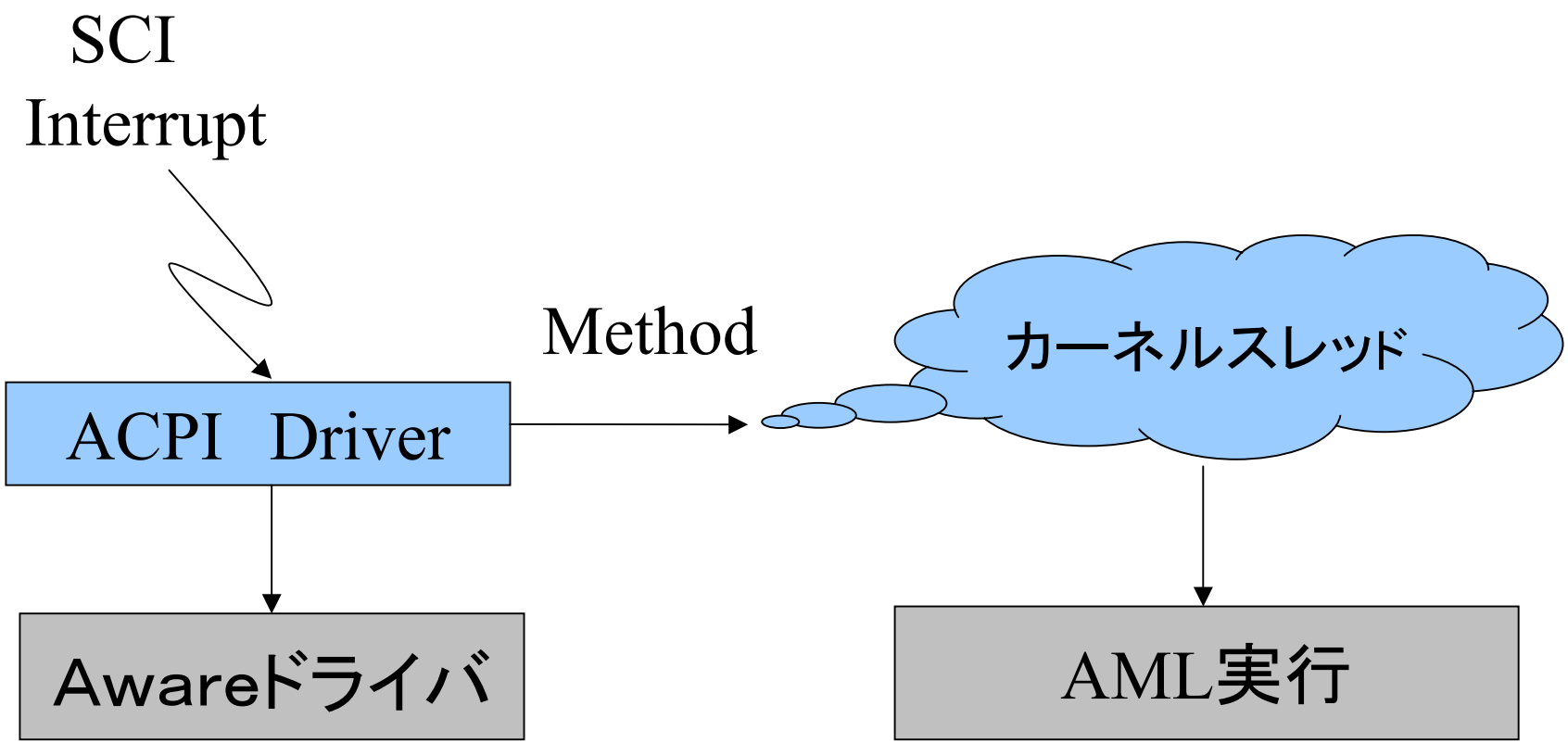


acpi_gbl_gpe_number_to_index[]

acpi_gbl_gpe_number_info[]のインデックス値を求める変換テーブル。



GPE処理概要


SCI
Interrupt
Method
カーネルスレッド
ACPI Driver
Awareドライバ
AML実行

GPEイベント割り込み処理流れ2

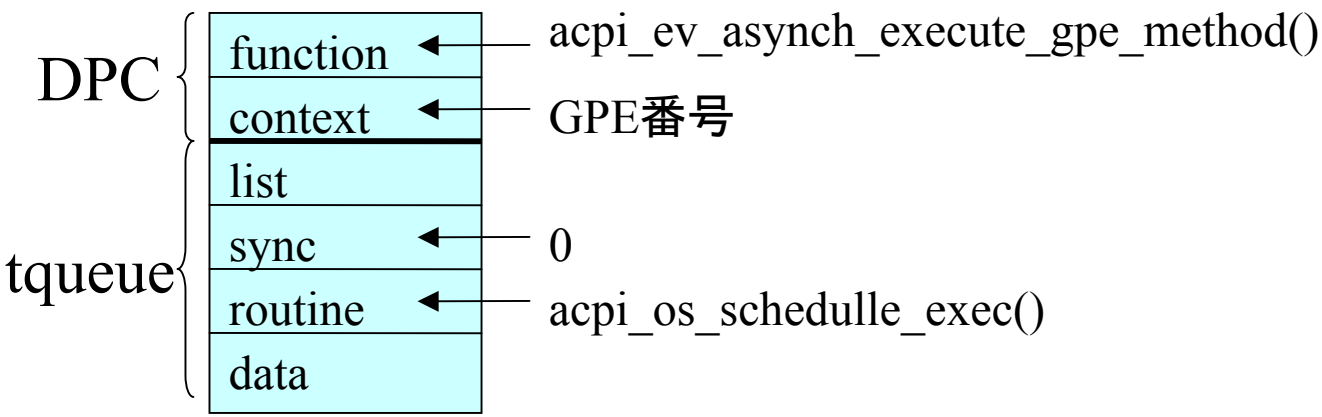
- acpi_ev_gpe_dispatch()
  - イベントのGPE情報(acpi_gbl_gpe_number_info[gpe番号])を得る。
  - GPE割り込みがエッジトリガの場合、イベントをクリアする。(acpi_hw_clear_gpe(gpe番号))
  - イベントに応じた処理を行う
    - ACPI-awareデバイスのイベントの場合、ACPI-awareドライバを呼び出す。
      gpe_info->handler(gpe_info->context);
    - それ以外の場合、制御methodを実行する(gpe_info->method_handle)
      - GPEメソッドを実行する。**acpi_os_queue_for_execution(xxxxx);**
  - ハンドラもメソッドも登録されていない場合は、使用できないのでGPEをDisableにする。
  - GPE割り込みがレベルトリガの場合、イベントをクリアする。
  - returnする

GPEイベント割り込み処理流れ3

acpi_os_queue_for_execution()

- ACPI_OS_DPC + Task queue構造体を作成する。まずは、メモリアロケートを行う。
- DPCに処理関数(acpi_ev_asynch_execute_gpe_method())、context(第三引数)を設定。
- Task Queueを初期化する。
- task queueをスケジューリング(登録)する。

  schedule_task(tq_struct)でtq_contextに登録しcontext_task_wqをWakeupする。
- returnする



GPEイベント割り込み処理流れ4

Task Queueの実行

- acpi_os_schedule_exec()
  - カーネルスレッド(acpi_os_queue_exec())を作成する。引数はDPC。

acpi_os_queue_exec()

- DPC(= acpi_ev_asynch_execute_gpe_method())を実行する
  - gpe_number_indexから、GPE情報(&acpi_gbl_gpe_number_info[gpe番号])を得る
  - 制御method(_Lxx, _Exx)を実行する

システム温度監視

- GPEを用いた温度監視
  - GPEイベントを起点とし、Core subsystemからThermalドライバへのNotify通知によって処理
- ポーリングによる温度監視
- APからの温度監視
  - Thremalドライバを経由した温度監視