




Wake up, FreeBSD! Implementing Modern Standby with SOix 🌙



Aymeric Wibo

Sponsored by the FreeBSD Foundation 🐉

About me

- OSs and graphics programming.
- Sponsored by the FreeBSD Foundation to work on this 
- Developer at Bnewable (energy storage solutions) 
- Spend most my time in a train 

Background

- S3 replaced by S0ix states in modern laptops for sleep.
- FreeBSD doesn't support S0ix yet.
- ∴ FreeBSD can't sleep on modern laptops.
- ∴ 😞

Previous work

- Ben Widawsky from Intel in 2018.
- [D17675](#) Suspend to idle support.
- [D17676](#) Emulated S3 with s0ix.

ACPI S3 🐌

- One of multiple global states, like S0 (active) and S5 (off).
- In S3, (almost) everything is off except for RAM.
- Heavy-handed approach: ask firmware to sleep, firmware sleeps.
- Slow to enter and exit.

S0ix

- Global state stays S0.
- Firmware decides when to enter S0ix state and turn off CPU.
- In theory, we "just" need to meet some device power constraints and idle the CPU.
- The end goal is S0i3 (saves the most power).

Terminology 🧐

- Shallower power/sleep state: closer to being on/awake.
- Deeper power/sleep state: closer to being off/asleep.
- In ACPI, shallower is a lower number (S0, on) and deeper is a higher number (S5, off).
- LPI state: low-power idle state.

Crash course on ACPI ✨

Crash course on ACPI

- Firmware exposes a bunch of information about hardware configuration through AML (ACPI machine language).
- Methods for telling devices what to do.
- E.g., decompiled AML (== ASL) for lid device (`acpi_lid`):

```
// Can decompile system's AML with: acpidump -dt
Device (_LID0) {
    Name (_HID, EisaId ("PNP0C0D") /* Lid Device */) // _HID: Hardware ID
    /* ... */
    Method (_LID, 0, NotSerialized) { // _LID: Lid Status
        Return (ESWL) // \_SB_.ESWL, see 19.6.47 in ACPI spec.
    }
}
```

Crash course on ACPI

```
/*
 * Evaluate _LID and check the return value, update lid status.
 *   Zero:           The lid is closed
 *   Non-zero:       The lid is open
 */
status = acpi_GetInteger(sc->lid_handle, "_LID", &lid_status);
if (ACPI_FAILURE(status))
    lid_status = 1;          /* assume lid is opened */
else
    lid_status = (lid_status != 0); /* range check value */
```



From `sys/dev/acpica/acpi_lid.c`.

SPMC (System Power Management
Controller) 🚓

SPMC (System Power Management Controller) 🚓

- **D48387** (`acpi_spmc`), **D48735**.
- Device specific method (`_DSM`): multiplexed vendor-defined function, `Arg0` is vendor-specific UUID:
 - `Arg2 = GET_DEVICE_CONSTRAINTS` : For each device, shallowest acceptable power state (min D-state). If a device violates its constraints, the system will not enter an LPI state!
 - `Arg2 = DISPLAY_OFF_NOTIF, ENTRY_NOTIF` : Tell FW we are entering modern standby.
 - `Arg2 = EXIT_NOTIF, DISPLAY_ON_NOTIF` : Tell FW we are exiting modern standby.
- Vendor-specific complications.

D-states

- Device power state.
- D0 (on), D1, D2, D3-ish (off).
- D3 split into D3hot  (off but with power) & D3cold  (off with no power).
- Not clear what D3 turns into: [acpica/acpica#993](#), [D48384](#).
- To get/set the D-state of a device, we need to introduce power resources.

Power resources ⚡

- A device is in the shallowest D-state which has all its power res. on.
- ⚠ If `_PR3` are off, the device is in `D3cold`.
- `_PSx` to set D-state, but power resources must be coherent!

```
PowerResource (P0NV, 0x00, 0x0000) {
    Method (_STA, 0, NotSerialized) { /* ... */ } // Status.
    Method (_ON, 0, NotSerialized) { /* ... */ } // Turn on.
    Method (_OFF, 0, NotSerialized) { /* ... */ } // Turn off.
}
Device (NVME) {
    Name (_PR0, Package (0x01) { P0NV }) // Power resources for D0.
    Name (_PR2, Package (0x01) { P0NV }) // Power resources for D2.
    Name (_PR3, Package (0x01) { P0NV }) // Power resources for D3hot.
    Method (_PS0, 0, NotSerialized) { /* ... */ } // Set to D0.
    Method (_PS3, 0, NotSerialized) { /* ... */ } // Set to D3.
}
```

Power resources ⚡

- [D48385](#) acpi_powerres: Fix turning off power resources on first D-state switch
- [D48386](#) acpi_powerres: `acpi_pwr_get_state` and getting initial D-state for device

Suspend-to-idle (S2idle) zzZ

Suspend-to-idle (S2idle)

- x86 `MWAIT` instruction, similar to `HLT`.
- Usually used in conjunction with `MONITOR`.
- Can be used to idle CPU until next interrupt:

```
mov eax, 0x30 ; C-state C4 (MWAIT_C4).  
mov ecx, 1    ; Break on interrupt, like hlt (MWAIT_INTRBREAK).  
mwait
```

- FreeBSD's `cpu_idle()` will use this when available.

Suspend-to-idle (S2idle)

- Since we wake from idle on interrupt, we should disable all interrupts except for wake interrupts.
- When the firmware has something to say, it sends a GPE (general purpose event) by triggering an SCI (system control interrupt).
- So, enable SCIs, interrupt 9.

```
register_t rflags = intr_disable(); // Save previous IF, run x86 cli.
intr_suspend(); // Stop interrupts from all PICs.
intr_enable_src(AcpiGbl_FADT.SciInterrupt); // Enable SCIs (interrupt 9).

cpu_idle(0); // Actually idle.

intr_resume(false); // Resume interrupts on all PICs.
intr_restore(rflags); // Restore IF.
```

Suspend-to-idle (S2idle)

- However, the firmware talks a LOT. Obviously, we don't want to wake every time the firmware has something to say...
- Normally, you can choose if a device can send GPEs to wake a system with `_DSW` (previously `_PSW`, see 7.3.1). But...

Suspend-to-idle (S2idle)

- Both important wake devices, such as our old friend `LID0`, and very talkative devices that we'd like to shut up, such as `BAT1`, are on the same device, and thus share a GPE number:

```
Device (EC0) { // The embedded controller.
    Name (_GPE, 0x0B) // GPE number.
    Device (LID0) { /* ... */ }
    Method (_Q01, 0, NotSerialized) { // GPE for lid device.
        P80H = 0x01
        Notify (LID0, 0x80) // Status change.
    }
    Method (_Q3C, 0, NotSerialized) { // VERY noisy GPE for battery (1 GPE/s).
        P80H = 0x3C
        Notify (BAT1, 0x80) // Status change.
    }
}
Device (BAT1) { /* ... */ }
```

Suspend-to-idle (S2idle)

- Solution? I don't know, the SPMC entry/exit notifications seem to help, but still being woken up by 1 battery GPE/min.
- Best effort: idle loop.
- Each time we break out of idle, check the wake reason and idle again ASAP if it isn't a real wake event.
- On AMD, the SMU needs to be hinted that we're entering/exiting sleep.
- [D48732](#), [D48734](#).

Debugging 

Debugging: LPIT/ `_LPI`

- Residency counters: count up how long we've been in a given LPI state.
- Intel devices have LPIT (LPI table), which has residency counters.
- ACPI defines `_LPI` which AMD (and ARM?) have.
- Residency counters are optional in `_LPI` however (8.4.3.3), and AMD doesn't populate them:

“ The register is optional. If the platform does not support it, then the following NULL register descriptor should be used: `ResourceTemplate()`

```
{Register {(SystemMemory, 0, 0, 0, 0)}}.
```

”

Debugging: AMD SMU

- System Management Unit.
- Runs PMFW, which is AMD's power management FW.
- Can send commands to it to get residency stats & hint we're entering/exiting sleep.
- [D48683](#), [D48714](#), [D48721](#).

Debugging: AMD SMU

- Not actually able to enter S0i3 on FreeBSD, though I am entering modern standby at least.
- On AMD, we need a USB4 connection manager to sleep the USB4 controllers, which FreeBSD doesn't have yet.
- More work needed!

✨ Demo ✨

Future

- Get to S0i3 :)))
- Give users the ability to define more complex wake rules.
- Hibernate (S4, suspend to disk) after suspended to memory for a certain amount of time.
- "Idleness determination" to automatically remove power from idle devices.


FreeBSD Foundation laptop project

Public issue tracker for all things FreeBSD laptop related:

<https://github.com/FreeBSDFoundation/proj-laptop/issues>

Please, do test this on your laptops and send me an email if something isn't working quite right!

Contact

- Have a beer! 
- FreeBSD email: obiwac@freebsd.org
- Website: <https://obiw.ac>
- GitHub: <https://github.com/obiwac>
- Discord: [@obiwac](#)