Embedded System Architecture

Software and hardware minimizing energy consumption

Conscious engineer protects the natur
Software and hardware minimizing energy consumption

- HW solutions
- SW solutions
  - μControllers
  - DSP processors
- HW solutions
- SW solutions
- Intel and AMD processors
- ACPI and other technologies
Software and hardware minimizing energy consumption

Minimizing power consumption – in practice – the action in order to reach a compromise between the energy consumed by the system, and its performance.
Software and hardware minimizing energy consumption

In many applications, it is required at the same time low power consumption and high performance, which is contradictory, forcing designers to seek compromise.

Energy demand should be considered at the concept stage of the system.
Software and hardware minimizing energy consumption

In many cases, wrong decisions will have irreversible effects:

System excessive energy consumption.

System performance below minimum acceptable level.
Software and hardware minimizing energy consumption

- System conception
- System design
- HW and SW tests
- SW tests
- HW and SW integration
- System Tests
- Production

Acceptance: TESTS PASS

System meets all requirements

Time
Software and hardware minimizing energy consumption

Haste is a bad counselor!
Software and hardware minimizing energy consumption - µControllers

Hardware minimizing energy consumption

Work with a supply voltage as low as possible - power consumption is proportional to the square of the supply voltage.

Consequence of the above

Lower supply voltage

Lower system clock frequency

Lower system performance
Software and hardware minimizing energy consumption - µControllers

Hardware minimizing energy consumption

Work with a system clock frequency as low as possible - power consumption is proportional to the system clock frequency.

Consequence of the above

- Lower system clock frequency
- Extended processing time
Software and hardware minimizing energy consumption - µControllers

Frequency versus supply voltage – AT Xmega family.

Figure 36-1. Maximum Frequency vs. $V_{CC}$.  

![Graph showing frequency versus supply voltage for the AT Xmega family with a shaded area indicating the safe operating range.](image-url)
Software and hardware minimizing energy consumption - µControllers

How to save energy?

By using of available power-saving modes, as follows:
- µController should spend as much time as possible in sleep modes.
- In Active mode µController should work with maximum performance.
- Active mode should be as short as possible.

It is a well known method.

Effectiveness of this method depends on:
- how the designer can deal with the effective use of power-saving modes and shorten the duration of the application with maximum performance.
Software and hardware minimizing energy consumption - μControllers

Estimating the energy demand of the system using power-saving modes designer has to take into account in the energy balance energy needed to wake up CPU (wake-up energy).

- Estimating excluding wake-up energy
- Estimating including wake-up energy
## Software and hardware minimizing energy consumption - μControllers

### TI MSP430 operating modes

<table>
<thead>
<tr>
<th>Operating Mode</th>
<th>Description</th>
<th>CPU (MCLK)</th>
<th>SMCLK</th>
<th>ACLK</th>
<th>RAM Retention</th>
<th>BOR</th>
<th>Self Wakeup</th>
<th>Interrupt Sources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>CPU, all clocks and peripherals available.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timers, ADC, DMA, UART, WDT, I/O, comparator, Ext. Interrupt, RTC, serial communications, other peripherals</td>
</tr>
<tr>
<td>LPM0</td>
<td>CPU is shutdown, peripheral clocks available.</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>Timers, ADC, DMA, UART, WDT, I/O, comparator, Ext. Interrupt, RTC, serial communications, other peripherals</td>
</tr>
<tr>
<td>LPM1</td>
<td>CPU is shutdown, peripheral clocks available. DCO is disabled and the DC generator can be disabled.</td>
<td>●</td>
<td></td>
<td></td>
<td>●</td>
<td>●</td>
<td></td>
<td>Timers, ADC, DMA, UART, WDT, I/O, comparator, Ext. Interrupt, RTC, serial communications, other peripherals</td>
</tr>
<tr>
<td>LPM2</td>
<td>CPU is shutdown, only one peripheral clock available. DC generator is enabled.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timers, ADC, DMA, UART, WDT, I/O, comparator, Ext. Interrupt, RTC, serial communications, other peripherals</td>
</tr>
<tr>
<td>LPM3</td>
<td>CPU is shutdown, only one peripheral clock available. DC generator is disabled.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Timers, ADC, DMA, UART, WDT, I/O, comparator, Ext. Interrupt, RTC, serial communications, other peripherals</td>
</tr>
<tr>
<td>LPM3.5</td>
<td>No RAM retention. RTC can be enabled. (MSP430F5xx generation only)</td>
<td>●</td>
<td></td>
<td>●</td>
<td>●</td>
<td>●</td>
<td></td>
<td>Ext. Interrupt, RTC</td>
</tr>
<tr>
<td>LPM4</td>
<td>CPU is shutdown and all clocks disabled.</td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td>●</td>
<td></td>
<td>Ext. Interrupt</td>
</tr>
<tr>
<td>LPM4.5</td>
<td>No RAM retention. RTC disabled. (MSP430F5xx generation only)</td>
<td></td>
<td></td>
<td>●</td>
<td></td>
<td></td>
<td></td>
<td>Ext. Interrupt</td>
</tr>
</tbody>
</table>
Software and hardware minimizing energy consumption - µControllers
Software and hardware minimizing energy consumption - μControllers

TI MSP430 operating modes

Use Interrupts & Maximize LPM3

170 μA

Active

Standby (LPM3)

0.4 μA

Active

Leave On the Slow Clock

- Low power clock and peripherals interrupt CPU only for processing

On-Demand CPU Clock

- DCO starts immediately
- CPU processes data and quickly returns to Low Power Mode

Source: https://www.slideshare.net/BenakPatel/msp430-low-power-modes
Software and hardware minimizing energy consumption - μControllers

TI MSP430 operating modes

MSP430™ Series Comparison

<table>
<thead>
<tr>
<th>Mode</th>
<th>G2xx</th>
<th>F5xx</th>
<th>FR57xx</th>
<th>FR58xx/FR59xx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Performance (max)</td>
<td>16 MHz</td>
<td>25 MHz</td>
<td>24 MHz (FRAM at 8MHz)</td>
<td>16 MHz (FRAM at 8MHz)</td>
</tr>
<tr>
<td>Flex Unified Memory</td>
<td>No</td>
<td>No</td>
<td>FRAM (16K)</td>
<td>FRAM (64K)</td>
</tr>
<tr>
<td>Active</td>
<td>AM</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>230 μA (1MHz)</td>
<td>180 μA/MHz</td>
<td>100 μA/MHz</td>
<td>&lt;100 μA/MHz</td>
</tr>
<tr>
<td>Standby</td>
<td>LPM3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.7 μA</td>
<td>1.9 μA</td>
<td>6.3 μA</td>
<td>0.7 μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2.1 μA</td>
<td>1.5 μA</td>
<td>0.4 μA</td>
</tr>
<tr>
<td>RTC</td>
<td>LPM3.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1 μA</td>
<td>1.1 μA</td>
<td>5.9 μA</td>
<td>0.6 μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2 μA</td>
<td>0.3 μA</td>
<td>0.1 μA</td>
</tr>
<tr>
<td>Off</td>
<td>LPM4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.1 μA</td>
<td>1.1 μA</td>
<td>5.9 μA</td>
<td>0.6 μA</td>
</tr>
<tr>
<td></td>
<td>LPM4.5</td>
<td></td>
<td></td>
<td>0.1 μA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.2 μA</td>
<td>0.3 μA</td>
<td></td>
</tr>
<tr>
<td>Wake-up from</td>
<td>Standby</td>
<td>3.5 μs</td>
<td>78 μs</td>
<td>&lt;10 μs</td>
</tr>
<tr>
<td></td>
<td></td>
<td>or 150 μs</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Off</td>
<td>2000 μs</td>
<td>310 μs</td>
<td>150 μs</td>
</tr>
</tbody>
</table>

Source: https://www.slideshare.net/BenakPatel/msp430-low-power-modes
Software and hardware minimizing energy consumption - μControllers

TI MSP430 operating modes

Entering Low Power Modes

<table>
<thead>
<tr>
<th>Enter LPMx</th>
<th>C Compiler Intrinsic</th>
<th>Writing to SR with Intrinsic</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPM0</td>
<td>_low_power_mode_0(); _bis_SR_register( GIE + LPM0_bits );</td>
<td></td>
</tr>
<tr>
<td>LPM1</td>
<td>_low_power_mode_1(); _bis_SR_register( GIE + LPM1_bits );</td>
<td></td>
</tr>
<tr>
<td>LPM2</td>
<td>_low_power_mode_2(); _bis_SR_register( GIE + LPM2_bits );</td>
<td></td>
</tr>
<tr>
<td>LPM3</td>
<td>_low_power_mode_3(); _bis_SR_register( GIE + LPM3_bits );</td>
<td></td>
</tr>
<tr>
<td>LPM4</td>
<td>_low_power_mode_4(); _bis_SR_register( GIE + LPM4_bits );</td>
<td></td>
</tr>
</tbody>
</table>

- As written, both intrinsic functions enable interrupts and the associated low-power mode
- bis (and bic) instructions mimic assembly language:
  - bis = bit set
  - bic = bit clear
- bis/bic intrinsics allows greater flexibility in selecting bits to set/clear

Source: https://www.slideshare.net/BenakPatel/msp430-low-power-modes

How do I exit LPMx?
Can I automatically reenter LPM?
Software and hardware minimizing energy consumption - µControllers

TI MSP430 operating modes

Automatically Re-entering LPM (after ISR)

```c
main()
{
    initGpio();
    initClocks();
    initTimers();
    _low_power_mode_3();
    while(1);
}
```

- Executing LPM3 function puts the processor standby
- Unless an interrupt occurs, CPU will stay asleep

When an interrupt wakes the CPU...

Source: https://www.slideshare.net/BenakPatel/msp430-low-power-modes
Software and hardware minimizing energy consumption - μControllers

TI MSP430 operating modes

Automatically Re-entering LPM (after ISR)

```c
main()
{
    initGpio();
    initClocks();
    initTimers();
    _low_power_mode_3();
    //while(1);
}

#pragma vector = TIMER1_A0
__interrupt ISR()
{
    GPIO_toggleOutputOnPin()
    // Return from interrupt (RETI)
}
```

- Executing LPM3 function puts the processor standby
- Unless an interrupt occurs, CPU will stay asleep
- No while{} loop is needed

- An interrupt wakes the CPU
- Status Register (SR) is saved to stack (including the LPM setting)
- Exiting ISR routine:
  - Compiler uses RETI instruction which restores SR from stack
  - Restoring SR places CPU back into low-power mode

But what if I want to leave the LPM after an interrupt?

Source: https://www.slideshare.net/BenakPatel/msp430-low-power-modes
Software and hardware minimizing energy consumption - μControllers

TI MSP430 operating modes

Leaving LPM (after ISR)

```c
main()
{
    initGpio();
    initClocks();
    initTimers();
    while(1){
        __low_power_mode_3();
        filter();
    }
}

#pragma vector = TIMER1_A0
__interrupt ISR()
{
    getSample();
    __low_power_mode_off_on_exit();
} // Return from interrupt (RETI)
```

- Executing LPM3 function puts the processor standby
- Unless an interrupt occurs, CPU will stay asleep
- Since ISR exits from LPM, we need additional code (such as a while{} loop)

- An interrupt wakes the CPU
- Status Register (SR) is saved to stack (including LPM bits)
- Exiting ISR routine:
  - ‘exit’ fcn modifies saved SR (clearing LPM) before restore
  - RETI instruction restores SR from stack
  - With LPM “off”, CPU returns to instruction after LPM intrinsic; e.g. filter()

Source: https://www.slideshare.net/BenakPatel/msp430-low-power-modes
# Software and hardware minimizing energy consumption - μControllers

## ATMEL XMEGA operating modes

<table>
<thead>
<tr>
<th>Sleep modes</th>
<th>Active clock domain</th>
<th>Oscillators</th>
<th>Wake-up sources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU clock</td>
<td>Peripheral clock</td>
<td>RTC clock source</td>
</tr>
<tr>
<td>Idle</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Power down</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power save</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Standby</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Extended standby</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: 1. Only from internal 8MHz oscillator in low power mode.
Software and hardware minimizing energy consumption - μControllers

ATMEL XMEGA Power Reduction Registers

The **power reduction (PR) registers** provide a method to stop the clock to individual peripherals. When this is done, the current state of the peripheral is frozen and the associated I/O registers cannot be read or written. Resources used by the peripheral will remain occupied; hence, the peripheral should be disabled before stopping the clock. Enabling the clock to a peripheral again puts the peripheral in the same state as before it was stopped. This can be used in idle mode and active modes to reduce the overall power consumption. In all other sleep modes, the peripheral clock is already stopped.

Not all devices have all the peripherals associated with a bit in the power reduction registers. Setting a power reduction bit for a peripheral that is not available will have no effect.
Software and hardware minimizing energy consumption - µControllers

ATMEL XMEGA enabling sleep mode

```c
#include <avr/io.h>
#include <avr/sleep.h>

int main(void)
{
    set_sleep_mode(CURRENT_SLEEP_MODE);
    sleep_enable();
    sleep_cpu();
    while(1)
    {
    }
}
```
Software and hardware minimizing energy consumption - μControllers

STMicroelectronics STM32L1 operating modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Power consumption</th>
<th>CPU</th>
<th>Flash / EEPROM</th>
<th>RAM</th>
<th>DMA &amp; Periph</th>
<th>Clock</th>
<th>LCD</th>
<th>RTC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleep</td>
<td>100µA/MHz (R1)</td>
<td>NO</td>
<td>On</td>
<td>On</td>
<td>Active</td>
<td>Any</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low power run</td>
<td>10,4µA Flash OFF, 32 kHz</td>
<td>YES</td>
<td>On or Off</td>
<td>On</td>
<td>Active</td>
<td>MSI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Low power sleep</td>
<td>5,1µA (periph. Off)</td>
<td>NO</td>
<td>Off</td>
<td>On</td>
<td>Active</td>
<td>MSI</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stop with RTC</td>
<td>1,3µA (1,8V)</td>
<td>NO</td>
<td>Off</td>
<td>On</td>
<td>Stopped</td>
<td>LSE, LSI</td>
<td></td>
<td>Enable</td>
</tr>
<tr>
<td>Stop</td>
<td>500nA</td>
<td>NO</td>
<td>Off</td>
<td>On</td>
<td>Stopped</td>
<td>LSE, LSI</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>Standby with RTC</td>
<td>1µA (1,8V)</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>LSE, LSI</td>
<td>Off</td>
<td>Off</td>
</tr>
<tr>
<td>Standby</td>
<td>270nA</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>Off</td>
<td>LSE, LSI</td>
<td>Off</td>
<td>Off</td>
</tr>
</tbody>
</table>
Software and hardware minimizing energy consumption - μControllers

Other possibilities of minimizing power consumption.

- Effective usage of the Interrupt system
- Effective usage of the Event system
- Data transfer via DMA channels

Lower power consumption
Software and hardware minimizing energy consumption - μControllers

Software compilation with code optimization:

- optimization option – max. speed:
  - Cost:
    - larger code size;
  - Benefits:
    - shorter work time with max. performance;
    - larger number of the simpler instructions to execute.

<table>
<thead>
<tr>
<th>Compilation option</th>
<th>Code size</th>
<th>Energy consumed within 1 sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max. speed</td>
<td>5456b (+38%)</td>
<td>41 μJ</td>
</tr>
<tr>
<td>Min. size</td>
<td>3944b</td>
<td>77.6 μJ (+92%)</td>
</tr>
</tbody>
</table>
Software and hardware minimizing energy consumption - µControllers

Other exemplary methods:
- Repeated explicit substitutions – reduce the number of jump instructions;
- Replacement of a single complex instruction set of simple instructions e.g. modulo instruction (may take from 2 to 12 clock cycles) – may be replaced by a simple control of the variable state.
Software and hardware minimizing energy consumption – DSP processors

- Frequency scaling.
- Supply voltage scaling.
- Usage of the power-saving modes:
  - Power STANDBY mode;
  - Power SLEEP mode;
  - Power DEEPSLEEP mode.
- Proper use of the above mentioned options.
Software and hardware minimizing energy consumption – DSP processors

- Effective usage of the Interrupt system;
- Data transfer via DMA channels;
- Effective usage of the Event system;
- Code optimization;
- FFT hardware acceleration;
Software and hardware minimizing energy consumption – TI DSP processors C5XXX

- Power Scaling Library:
  - Scaling operation to scale frequency and voltage or frequency only;
  - Query operations that return current frequency and voltage settings;
  - Query operations that return available frequencies settings and the required voltage settings for those frequencies;
  - Query operation that returns the latencies associated with a scaling operation;
  - Callbacks to user code before and after scaling operations. These callbacks will enable users to perform any necessary peripheral modifications that may be required as a result of the upcoming/just completed scaling operation.
Software and hardware minimizing energy consumption – TI DSP processors C5XXX

![Graph showing power consumption and time](image)

**Figure 1. Effects of Scaling on Power Consumption**
Software and hardware minimizing energy consumption – TI DSP processors

The DSP/BIOS Power Manager, PWRM – module that lets you reduce the power consumption of your application.
The PWRM module provides the following capabilities:
- Resource Tracking.
- Scaling Voltage and Frequency.
- Using Sleep Modes.
- Coordinating Sleep and Scaling.

’C55x PWRM additional capabilities:
- Idling Clock Domains.
- Saving Power at Boot Time.
- DSP Device Initialization.
Software and hardware minimizing energy consumption – TI DSP processors

Figure 1. PWRM in an Application’s Architecture
Software and hardware minimizing energy consumption – TI DSP processors

\[
P_{\text{total}} = P_{\text{active}} + P_{\text{static}}
\]

\[
P_{\text{active}} \approx C \times F \times V^2 \times N_{\text{sw}}
\]

\[
P_{\text{static}} \approx V \times I_q
\]

- \(C\) – dynamic capacitance;
- \(F\) – switching frequency;
- \(V\) – supply voltage;
- \(N_{\text{sw}}\) – number of switching;
- \(V\) – supply voltage;
- \(I_q\) – leakage current;
Software and hardware minimizing energy consumption – Analog Devices DSP processors

*Dynamic Power Management Controller* – consists of two main components which allows flexibility in managing power dissipation on the processor:

- Operating Modes;
- Peripheral Clocking;
- Dynamic Voltage Control.

Operating modes:
- Full-on mode;
- Active mode;
- Sleep mode;
- Deep Sleep mode.
Software and hardware minimizing energy consumption – Analog Devices DSP processors

Effects of V and F changes on power consumption Blackfin™ Processors.
Software and hardware minimizing energy consumption – Analog Devices DSP processors

Savings in power consumption can be modeled by the following equation:

\[
\frac{P_R}{P_N} = \left( \frac{F_{CR}}{F_{CN}} \right) \times \left( \frac{V_{DDR}}{V_{DDN}} \right)^2 \times \left( \frac{T_{FR}}{T_{FN}} \right)
\]

where:

- \(P_R/P_N\) is the ratio of reduced power to nominal power
- \(F_{CN}\) is the nominal core clock frequency
- \(F_{CR}\) is the reduced core clock frequency
- \(V_{DDN}\) is the nominal internal supply voltage
- \(V_{DDR}\) is the reduced internal supply voltage
- \(T_{FR}\) is the duration running at \(F_{CR}\)
- \(T_{FN}\) is the duration running at \(F_{CN}\)
Software and hardware minimizing energy consumption – Analog Devices DSP processors

Example:

\[
\frac{P_R}{P_N} = \left( \frac{100}{300} \right) \times \left( \frac{1.0}{1.5} \right)^2 \times \left( \frac{3}{1} \right) = 44\%, \text{ savings } 56\%
\]

- \( F_{CN} = 300 \text{ MHz} \)
- \( F_{CR} = 100 \text{ MHz} \)
- \( V_{DDN} = 1.5 \text{ V} \)
- \( V_{DDR} = 1.0 \text{ V} \)
- \( T_{FR} = 3 \)
- \( T_{FN} = 1 \)
ACPI - Advanced Configuration and Power Interface

The principal goals of ACPI and OSPM Operating System Power Management are to:

- Enable all computer systems to implement motherboard configuration and power management functions, using appropriate cost/function tradeoffs.
- Enhance power management functionality and robustness.
  - Power management policies too complicated to implement in a ROM BIOS can be implemented and supported in the OS, allowing inexpensive power managed hardware to support very elaborate power management policies.
  - Gathering power management information from users, applications, and the hardware together into the OS will enable better power management decisions and execution.
  - Unification of power management algorithms in the OS will reduce conflicts between the firmware and OS and will enhance reliability.
- Facilitate and accelerate industry-wide implementation of power management.
- Create a robust interface for configuring motherboard devices.
ACPI - Advanced Configuration and Power Interface

The functional areas covered by the ACPI specification:

- System power management
- Device power management
- Processor power management
- Device and processor performance management
- Configuration / Plug and Play
- System Events
- Battery management
- Thermal management
- Embedded Controller
- SMBus Controller
ACPI - Advanced Configuration and Power Interface

**System power management**

- defines mechanisms for putting the computer as a whole in and out of system sleeping states
- provides a general mechanism for any device to wake up a computer.

**Device power management**

- Describe:
  - motherboard devices
  - their power states
  - the power planes the devices are connected to
  - controls for putting devices into different power states

- This enables the OS to put devices into low-power states based on application usage.
ACPI - Advanced Configuration and Power Interface

Processor power management

- While the OS is idle but not sleeping, it will use commands described by ACPI to put processors in low-power states.

Device and processor performance management

- While the OS is active, OSPM will transition devices and processors into different performance states, defined by ACPI, to achieve a desirable balance between performance and energy conservation goals as well as other environmental requirements (for example, visibility and acoustics).
ACPI - Advanced Configuration and Power Interface

Configuration / Plug and Play

- ACPI specifies:
  - information used to enumerate and configure motherboard devices. This information is arranged hierarchically so when events such as docking and undocking take place, the OS has precise, a priori knowledge of which devices are affected by the event.
ACPI - Advanced Configuration and Power Interface

System Events

- ACPI provides:
  - a general event mechanism that can be used for system events such as:
    - thermal events,
    - power management events,
    - docking, device insertion and removal, and so on.

This mechanism is very flexible in that, it does not define specifically how events are routed to the core logic chip set.
battery management policy moves from the APM BIOS to the ACPI OS.

- An ACPI compatible battery device needs either a Smart Battery subsystem interface, which is controlled by the OS directly through the embedded controller interface, or a Control Method Battery interface.
- A Control Method Battery interface is completely defined by ACPI Machine Language control methods, allowing an OEM to choose any type of the battery and any kind of communication interface supported by ACPI.
- The battery must comply with the requirements of its interface, as described either herein or in other applicable standards.
- The OS may choose to alter the behavior of the battery, for example, by adjusting the Low Battery or Battery Warning trip point.
- When there are multiple batteries present, the battery subsystem is not required to perform any synthesis of a “composite battery” from the data of the separate batteries. In cases where the battery subsystem does not synthesize a “composite battery” from the separate battery’s data, the OS must provide that synthesis.
Thermal management

- Since the OS controls the power and performance states of devices and processors, ACPI also addresses system thermal management. It provides a simple, scalable model that allows OEMs to define thermal zones, thermal indicators, and methods for cooling thermal zones.
ACPI - Advanced Configuration and Power Interface

Embedded Controller

- ACPI defines a standard hardware and software communications interface between an OS bus enumerator and an embedded controller. This allows any OS to provide a standard bus enumerator that can directly communicate with an embedded controller in the system, thus allowing other drivers within the system to communicate with and use the resources of system embedded controllers. This in turn enables the OEM to provide platform features that the OS and applications can use.
ACPI - Advanced Configuration and Power Interface

SMBus Controller

- ACPI defines a standard hardware and software communications interface between an OS bus driver and an SMBus Controller. This allows any OS to provide a standard bus driver that can directly communicate with SMBus devices in the system. This in turn enables the OEM to provide platform features that the OS and applications can use.
OSPM’s mission is to optimally configure the platform and to optimally manage the system’s power, performance, and thermal status given the user’s preferences and while supporting OS imposed Quality of Service (QOS) / usability goals. To achieve these goals, ACPI requires that once an ACPI compliant platform is in ACPI mode, the platform’s hardware, firmware, or other non-OS software must not manipulate the platform’s configuration, power, performance, and thermal control interfaces independently of OSPM. OSPM alone is responsible for coordinating the configuration, power management, performance management, and thermal control policy of the system.
ACPI - Advanced Configuration and Power Interface

Figure 3-2 Global System Power States and Transitions
ACPI - Global System State Definitions

Global system states \( (G_x) \) states apply to the entire system and are visible to the user.

Six principal criteria:
- Does application software run?
- What is the latency from external events to application response?
- What is the power consumption?
- Is an OS reboot required to return to a working state?
- Is it safe to disassemble the computer?
- Can the state be entered and exited electronically?
ACPI - Global System State Definitions

G3 Mechanical Off

It is implied by the entry of this off state through a mechanical means that no electrical current is running through the circuitry and that it can be worked on without damaging the hardware or endangering service personnel. The OS must be restarted to return to the Working state. No hardware context is retained. Except for the real-time clock, power consumption is zero.

G2/S5 Soft Off

A computer state where the computer consumes a minimal amount of power. No user mode or system mode code is run. This state requires a large latency in order to return to the Working state. The system’s context will not be preserved by the hardware. The system must be restarted to return to the Working state. It is not safe to disassemble the machine in this state.
ACPI - Global System State Definitions

G1 Sleeping

A computer state where the computer consumes a small amount of power, user mode threads are not being executed, and the system “appears” to be off (from an end user’s perspective, the display is off, and so on). Latency for returning to the Working state varies on the wake environment selected prior to entry of this state (for example, whether the system should answer phone calls). Work can be resumed without rebooting the OS because large elements of system context are saved by the hardware and the rest by system software. It is not safe to disassemble the machine in this state.

G0 Working

System dispatches user mode (application) threads and they execute. In this state, peripheral devices (peripherals) are having their power state changed dynamically. The user can select various performance/power characteristics of the system to have the software optimize for performance or battery life. The system responds to external events in real time. It is not safe to disassemble the machine in this state.
ACPI - Global System State Definitions

S4 Non-Volatile Sleep

A special global system state that allows system context to be saved and restored (relatively slowly) when power is lost to the motherboard. If the system has been commanded to enter S4, the OS will write all system context to a file on non-volatile storage media and leave appropriate context markers. The machine will then enter the S4 state. When the system leaves the Soft Off or Mechanical Off state, transitioning to Working (G0) and restarting the OS, a restore from a NVS file can occur. This will only happen if a valid non-volatile sleep data set is found, certain aspects of the configuration of the machine have not changed, and the user has not manually aborted the restore. If all these conditions are met, as part of the OS restarting, it will reload the system context and activate it. The net effect for the user is what looks like a resume from a Sleeping (G1) state (albeit slower). The aspects of the machine configuration that must not change include, but are not limited to, disk layout and memory size. It might be possible for the user to swap a PC Card or a Device Bay device, however.
ACPI - Device Power State Definitions

Device power states are states of particular devices; as such, they are generally not visible to the user. Device states apply to any device on any bus. Four principal criteria:

- Power consumption – How much power the device uses.
- Device context – How much of the context of the device is retained by the hardware. The OS is responsible for restoring any lost device context (this may be done by resetting the device).
- Device driver – What the device driver must do to restore the device to full on.
- Restore time - How long it takes to restore the device to full on.
ACPI - Device Power State Definitions

D3 (Off)

Power has been fully removed from the device. Also referred to as D3cold in this and other specs. All device context is lost when this state is entered, so the OS software will reinitialize the device when powering it back on. Since all device context and power are lost, devices in this state do not decode their address lines, and cannot be enumerated by software. Devices in this state have the longest restore times.
ACPI - Device Power State Definitions

D3hot

The meaning of the D3hot State is defined by each device class. In general, D3hot is expected to save as much power as possible without affecting PNP Enumeration. Devices in D3hot must have enough power to remain enumerable by software. For example, PCI Configuration space access and contents must operate as in shallower power states. Similarly, ACPI identification and configuration objects must operate as in shallower power states. Otherwise, no device functionality is supported, and Driver software is required to restore any lost context, or reinitialize the device, during its transition back to D0. Devices in this state can have long restore times. All classes of devices define this state.
ACPI - Device Power State Definitions

D2

The meaning of the D2 Device State is defined by each device class. Many device classes may not define D2. In general, D2 is expected to save more power and preserve less device context than D1 or D0. Buses in D2 may cause the device to lose some context (for example, by reducing power on the bus, thus forcing the device to turn off some of its functions).

D1

The meaning of the D1 Device State is defined by each device class. Many device classes may not define D1. In general, D1 is expected to save less power and preserve more device context than D2.
ACPI - Device Power State Definitions

D0 (Fully-On)

This state is assumed to be the highest level of power consumption. The device is completely active and responsive, and is expected to remember all relevant context continuously.
Device performance states (Px states) are power consumption and capability states within the active (D0) device power state. Performance states allow OSPM to make tradeoffs between performance and energy conservation. Device performance states have the greatest impact when the implementation is such that the states invoke different device efficiency levels as opposed to a linear scaling of performance and energy consumption. Since performance state transitions occur in the active device states, care must be taken to ensure that performance state transitions do not adversely impact the system.
S1-S4 are types of sleeping states within the global system state, G1. S5 is a soft-off state associated with the G2 system state.

**S1 Sleeping State**

The S1 sleeping state is a low wake latency sleeping state. In this state, no system context is lost (CPU or chip set) and hardware maintains all system context.

**S2 Sleeping State**

The S2 sleeping state is a low wake latency sleeping state. This state is similar to the S1 sleeping state except that the CPU and system cache context is lost (the OS is responsible for maintaining the caches and CPU context). Control starts from the processor’s reset vector after the wake event.
ACPI - Sleeping and Soft-off State Definitions

**S3 Sleeping State**

The S3 sleeping state is a low wake latency sleeping state where all system context is lost except system memory. CPU, cache, and chip set context are lost in this state. Hardware maintains memory context and restores some CPU and L2 configuration context. Control starts from the processor’s reset vector after the wake event.

**S4 Sleeping State**

The S4 sleeping state is the lowest power, longest wake latency sleeping state supported by ACPI. In order to reduce power to a minimum, it is assumed that the hardware platform has powered off all devices. Platform context is maintained.
S5 Soft Off State

The S5 state is similar to the S4 state except that the OS does not save any context. The system is in the “soft” off state and requires a complete boot when it wakes. Software uses a different state value to distinguish between the S5 state and the S4 state to allow for initial boot operations within the BIOS to distinguish whether or not the boot is going to wake from a saved memory image.
ACPI - Processor Power State Definitions

Processor power states (Cx states) are processor power consumption and thermal management states within the global working state, G0.

C0 Processor Power State
While the processor is in this state, it executes instructions.

C1 Processor Power State
This processor power state has the lowest latency. The hardware latency in this state must be low enough that the operating software does not consider the latency aspect of the state when deciding whether to use it. Aside from putting the processor in a non-executing power state, this state has no other software-visible effects.
ACPI - Processor Power State Definitions

**C2 Processor Power State**
The C2 state offers improved power savings over the C1 state. The worst-case hardware latency for this state is provided via the ACPI system firmware and the operating software can use this information to determine when the C1 state should be used instead of the C2 state. Aside from putting the processor in a non-executing power state, this state has no other software-visible effects.

**C3 Processor Power State**
The C3 state offers improved power savings over the C1 and C2 states. The worst-case hardware latency for this state is provided via the ACPI system firmware and the operating software can use this information to determine when the C2 state should be used instead of the C3 state. While in the C3 state, the processor’s caches maintain state but ignore any snoops. The operating software is responsible for ensuring that the caches maintain coherency.
ACPI - Device and Processor Performance State Definitions

Device and Processor performance states (Px states) are power consumption and capability states within the active/executing states, C0 for processors and D0 for devices.

**P0 Performance State**
While a device or processor is in this state, it uses its maximum performance capability and may consume maximum power.

**P1 Performance State**
In this performance power state, the performance capability of a device or processor is limited below its maximum and consumes less than maximum power.

**Pn Performance State**
In this performance state, the performance capability of a device or processor is at its minimum level and consumes minimal power while remaining in an active state. State n is a maximum number and is processor or device dependent. Processors and devices may define support for an arbitrary number of performance states not to exceed 255.
ACPI - Advanced Configuration and Power Interface - Definitions

**ACPI Machine Language (AML)** - Pseudo-code for a virtual machine supported by an ACPI-compatible OS and in which ACPI control methods and objects are written.

**Embedded Controller** - The general class of microcontrollers used to support OEM-specific implementations, mainly in mobile environments. The ACPI specification supports embedded controllers in any platform design, as long as the microcontroller conforms to one of the models described in this section. The embedded controller performs complex low-level functions through a simple interface to the host microprocessor(s).

**Embedded Controller Interface** - A standard hardware and software communications interface between an OS driver and an embedded controller. This allows any OS to provide a standard driver that can directly communicate with an embedded controller in the system, thus allowing other drivers within the system to communicate with and use the resources of system embedded controllers (for example, Smart Battery and AML code). This in turn enables the OEM to provide platform features that the OS and applications can use.
ACPI - Advanced Configuration and Power Interface - Definitions

**Power Management** – Mechanisms in software and hardware to minimize system power consumption, manage system thermal limits, and maximize system battery life. Power management involves trade-offs among system speed, noise, battery life, processing speed, and alternating current (AC) power consumption. Power management is required for some system functions, such as appliance (for example, answering machine, furnace control) operations.

**Smart Battery Subsystem** – A battery subsystem that conforms to the following specifications: Smart Battery and either Smart Battery System Manager or Smart Battery Charger and Selector — and the additional ACPI requirements.

**System Management Bus (SMBus)** – A two-wire interface based upon the I²C protocol. The SMBus is a low-speed bus that provides positive addressing for devices, as well as bus arbitration.

**SMBus Interface** - A standard hardware and software communications interface between an OS bus driver and an SMBus controller.
**ACPI - Advanced Configuration and Power Interface - Definitions**

**Thermal States** - Thermal states represent different operating environment temperatures within thermal zones of a system. A system can have one or more thermal zones; each thermal zone is the volume of space around a particular temperature-sensing device. The transitions from one thermal state to another are marked by trip points, which are implemented to generate an SCI when the temperature in a thermal zone moves above or below the trip point temperature.
SW and HW minimizing energy consumption – INTEL processors

Various power states:
- System Sleep States (S-States)
- Processor Power States (C-States)
- Processor Performance Power States (P-States)

Power Management Technologies:
- C-State Auto Demotion
- Intel® Turbo Boost Technology
- Virtualization Power Management
- Memory Power Management
Sprzętowo-programowa minimalizacja poboru energii – procesory firmy INTEL

Advanced Configuration and Power Interface (ACPI) Supported States

- **P-States** are a sub-state of the C0 state
  - Offer reduced power consumption while the processor is executing code.

- **C-States** are a sub-state of the S0 state
  - Offer reduced power consumption while the system is fully on.
SW and HW minimizing energy consumption – INTEL processors

System Sleeping Power States (S-States) – S-States allow the system to save a power when not being used.

- **S0 – Full on**: Processor Operating. Individual devices may be shut down or be placed into lower power states to save power.
- **S3 – Suspend-to-RAM**: System context maintained in DRAM. Power shut off to non-critical circuits. Memory is retained and refreshes continue. All clocks are stopped except RTC clock.
- **S4 – Suspend-to-Disk**: System context maintained on disk. All power is then shut off except for that needed to resume.
SW and HW minimizing energy consumption – INTEL processors

- **S5 – Soft off**: System context not maintained. All power is shut off except for that needed to restart. A full boot is required when waking.

S-States reduce power consumption significantly. Latency back to S0 is in the order of seconds.

Wake event can be triggered by:
- a motion sensor;
- remotely using Intel® Active Management Technology (Intel® AMT);
- a timer or another activity (LAN or GPIO) pins.
Processor Power and Performance Power States

Processor Power States (C-States):
- Reduces power consumption by sleeping the processor when it doesn’t have code to execute.
- Entry and Exit delays are much smaller compared to S-States.

Processor Performance Power States (P-States):
Enhanced Intel SpeedStep® Technology
- Reduces power consumption without preventing the processor from executing code.
SW and HW minimizing energy consumption – INTEL processors

Processor C-States:
- C-States ensure lower processor power during idle light workloads;
- C-State limits can be set by BIOS;
- A processor can go into sleep states several thousand times per second;
- OS controls the C-states in its idle process.
SW and HW minimizing energy consumption – INTEL processors

Processor C States

<table>
<thead>
<tr>
<th>Active state</th>
<th>Sleep states</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C0</td>
</tr>
<tr>
<td>Operating</td>
<td></td>
</tr>
</tbody>
</table>

- Core clock
- PLL
- Core caches
- Shared cache
- Wakeup time*
- Core Idle power*

* Rough approximation
SW and HW minimizing energy consumption – INTEL processors

C-State Auto Demotion:

In general, Latency and Energy entry/exit costs increase with deeper C-States.
C-State Auto Demotion prevents unnecessary excursions into C6 & C3 states to improve latency.
Two C-State auto-demotion options.

Requests for C6 or C3 will route to C3 or C1 until sufficient residency is established.

Decision to demote is based on each core’s immediate residency history.

This feature is disabled by default.

See your BIOS vendor to enable this feature.
**SW and HW minimizing energy consumption – INTEL processors**

**Processor Performance Power States (P-States):**

- **P0**: Processor consumes max power and is at max performance.
- **P1**: Processor consumes less power and performance capabilities are limited below max.
- **Pn**: Performance is at minimal level and lowest power consumption. n must not exceed 255.

ACPI Spec Rev. 6.0 Defined
SW and HW minimizing energy consumption – INTEL processors

Multiple voltage and frequency operating points:

- Software controlled by writing to MSRs - Model Specific Register;
- The voltage is optimized based on the selected frequency and number of active processor cores;
- All active processor cores share the same frequency and voltage.

Number of supported states is processor dependent.
P-states (Where are they useful?):

- P-States are useful when the system runs non-critical workloads that don’t require higher performance.
- P States are useful when the system in power sensitive markets that don’t care so much about performance. i.e longer battery life.
- P States is useful under interactive modes where the system is waiting for user inputs.
- When I/O or memory is throttled.
## SW and HW minimizing energy consumption – INTEL processors

The Aggressiveness of P states.

Set via OS:

- **Linux** – P States can be configured based on the end users using governors:
  - “On Demand” (Preferred) – can be customized;
  - Performance – Aggressive Performance;
  - Power save – Aggressive Power;
  - User space- User can set upper and lower limits.

- **Windows** – has power management schemes to set the values:
  - Balanced;
  - Max Battery;
  - Max Performance;
  - Custom.
**SW and HW minimizing energy consumption – INTEL processors**

Virtualization Power Management - Workload consolidation during off peak usage.
Memory Power Management.
Intel® Turbo Boost Technology.

<table>
<thead>
<tr>
<th>Intel® Core™ 2 Duo Processor</th>
<th>Intel® Core™ i7 Processor</th>
<th>Intel Core i7 Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Single core CPU Turbo</strong></td>
<td><strong>Multi-core CPU Turbo</strong></td>
<td><strong>Power Sharing, Graphics Turbo</strong></td>
</tr>
<tr>
<td>+1 bin (single core turbo)</td>
<td>+3-4 bins* (2 core turbo)</td>
<td></td>
</tr>
<tr>
<td>Sleep</td>
<td>Awake: Low CPU activity</td>
<td></td>
</tr>
</tbody>
</table>

**Scenario 1**
- CPU Intensive Load

**Scenario 2**
- Gfx Intensive Load
- Dynamically trade TDP budget

---

Embedded Systems Architecture
SW and HW minimizing energy consumption – AMD processors

AMD-P technologies:

Enhanced AMD PowerNow!™ Technology – for reduced power consumption by the entire processor. Native quad-core technology enables enhanced power management across all four cores.

- High Performance mode;
- Power-Saver mode;
- Automatic mode.
Independent Dynamic Core Technology – allows each core to vary its frequency, based on the specific needs of the system. This allows for more precise power management to reduce data center energy consumption and thereby reduce total cost of ownership (TCO).

Dual Dynamic Power Management™ - allows each processor to maximize the power-saving benefits of Enhanced AMD PowerNow! technology without compromising performance. Dual Dynamic Power Management can reduce idle power consumption and allow for per-processor power management in multi-socket systems to decrease power consumption.
SW and HW minimizing energy consumption – AMD processors

**AMD CoolCore™ Technology** – To reduce power consumption within each core AMD CoolCore Technology evaluates which parts of the die – the cores, the memory, or both – are needed to support currently running applications. It can cut power to unused transistor areas to reduce power consumption and lower heat generation.

**AMD Smart Fetch Technology** – AMD Smart Fetch Technology helps reduce power consumption by allowing idle cores to enter a “halt” state, causing them to draw even less power during processing idle times.
SW and HW minimizing energy consumption – AMD processors

**Integrated DDR3 DRAM Memory Controller:** low-power memory to help reduce power consumption. AMD’s integrated memory controller works with high bandwidth, energy-efficient DDR3 memory, both standard power 1.5v and low voltage 1.35v memory. It incorporates memory RAS for increased fault tolerance to help reduce system downtime and increase system reliability.
SW and HW minimizing energy consumption – links and references

- Advanced Configuration and Power Interface Specification, Revision 6.0 April, 2015 http://www.acpi.info/
- Intel® Core™2 Processor Family Power States (P-States) Results Graph, Source: http://download.intel.com/design/mobile/datasheets/32012001.pdf