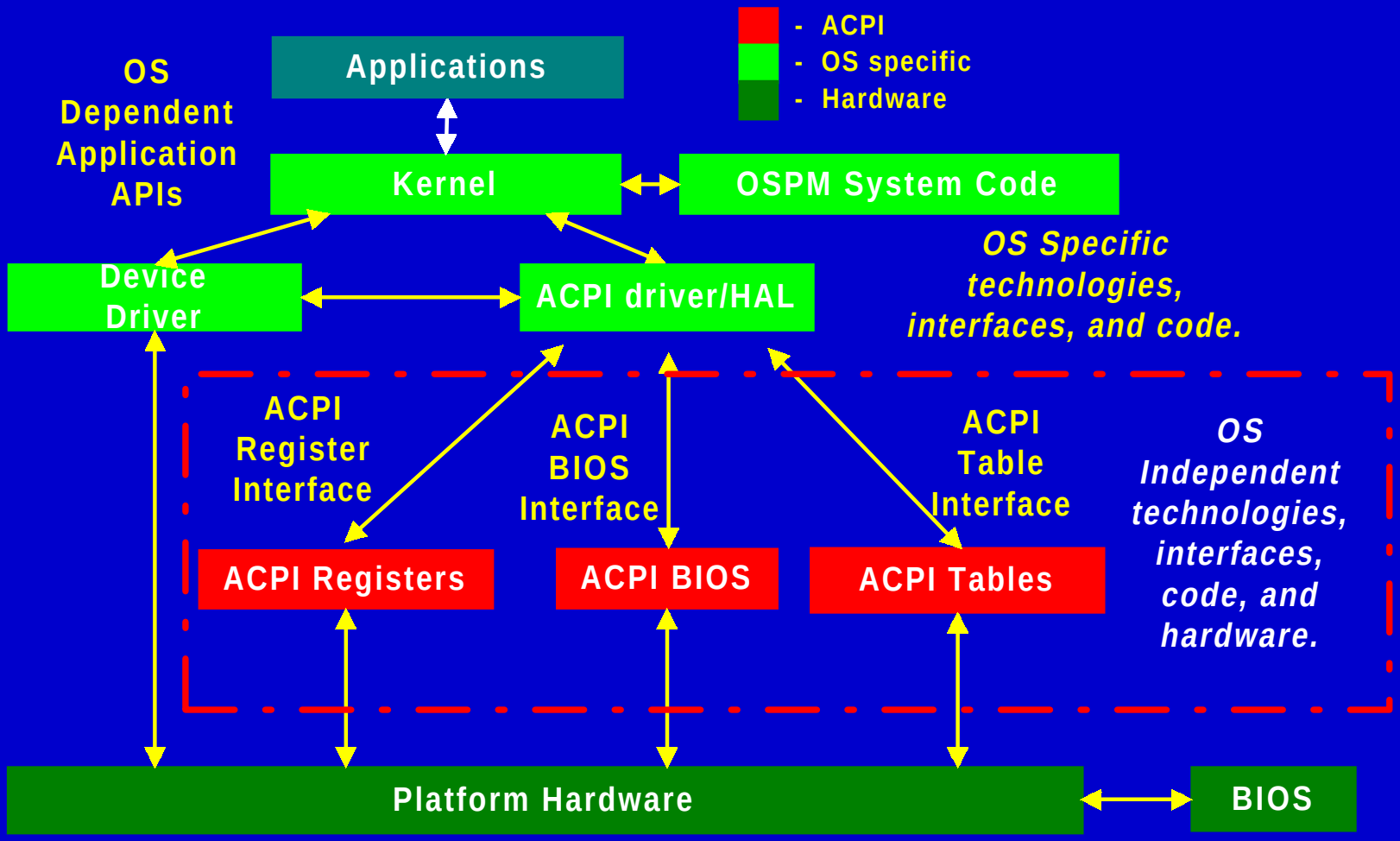
ACPI in Win 98 and NT 5.0

Phil Mummah
Intel Corporation

Agenda

- ACPI tables & code
- Device power states
- Sleep states
- Battery Intro

ACPI Interface



ACPI Tables

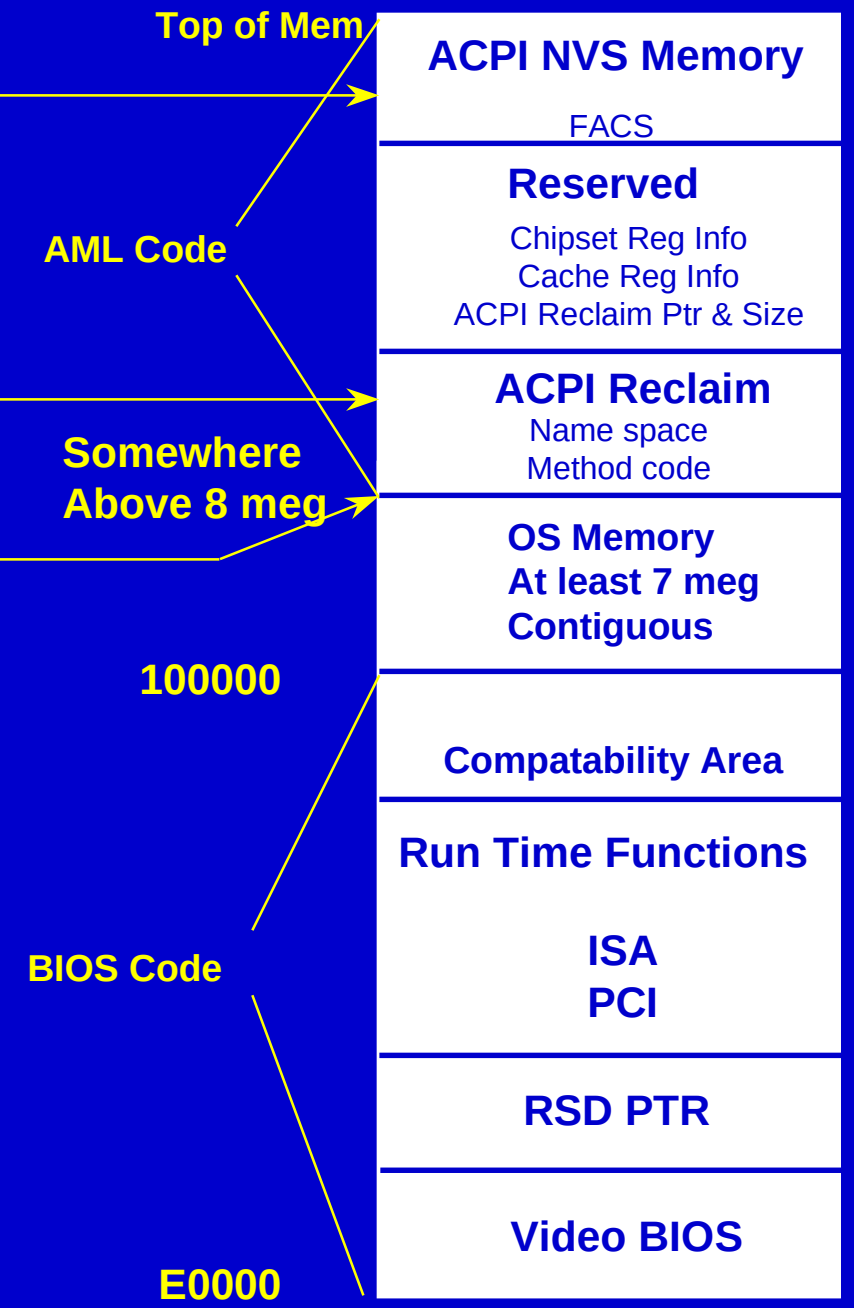
- Describes the machine to the OS
  - System capabilities
  - Controls to hand off to ACPI mode
  - Locations of register blocks
  - Real-time clock
  - Firmware handshaking

Memory Map After Boot

Int 15 Func E820 return value 4:
Returns start and size of NVS

Int 15 Func E820 return value 3:
Returns start and size of Reclaim

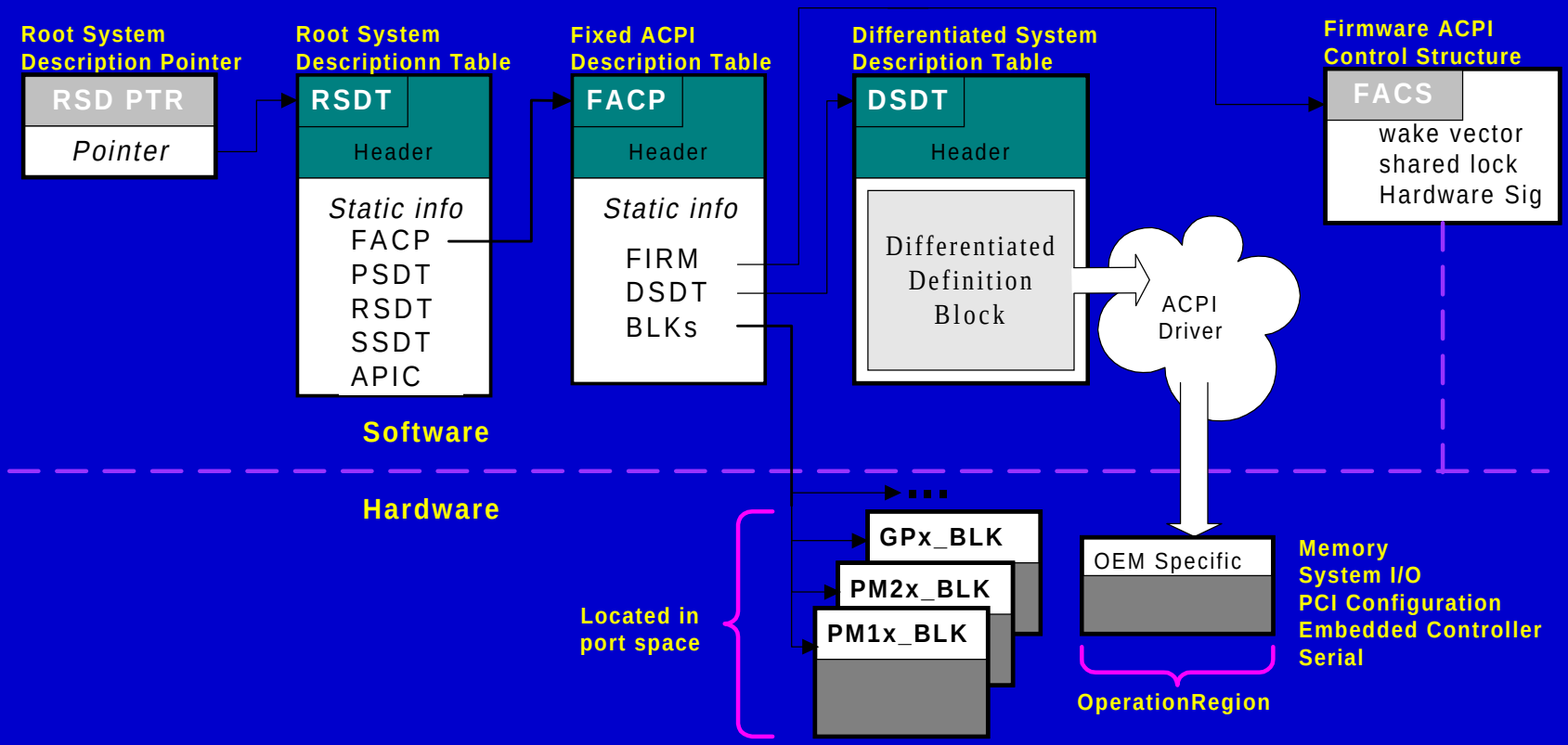
Int 15 functions 88 & E801
Returns size from 0 to start of ACPI memory.



System Description Tables

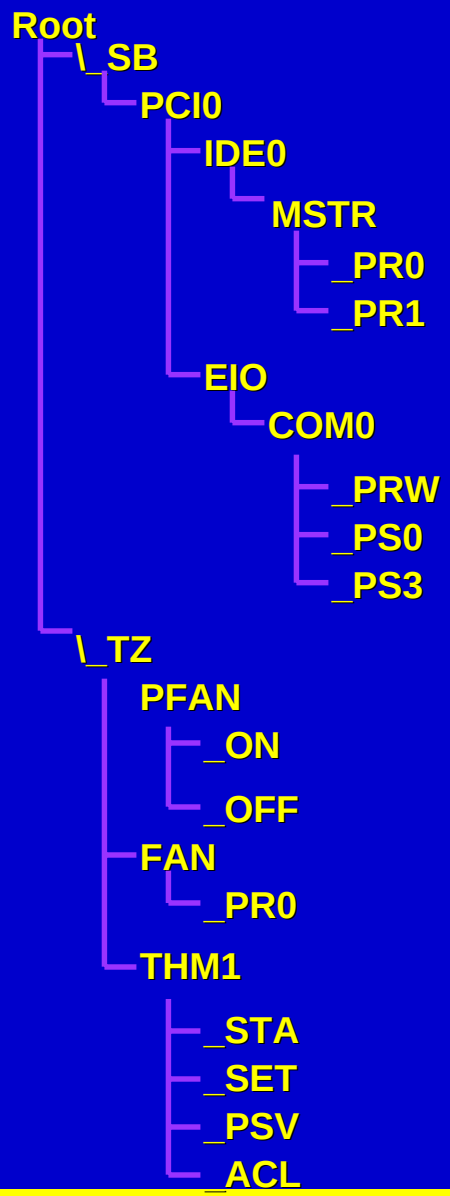
- Hierarchical description of the devices and controls on the motherboard
- Dynamic: additional blocks can be loaded/unloaded on the fly
- Written in ASL, encoded in AML
  - AML = ACPI Machine Language
  - ASL = ACPI Source Language

FACP Table-Putting It Together



- FACP contains all the pointers to fixed HW
  - Pointers to the different register blocks
    - Contains flags to configure fixed H/W
  - Points to DSDT (namespace)
  - Points to FACS (shared memory)

ACPI Namespace




```
Scope (\_SB) {
    Device(PCI0) {
        Device (IDE0) {
            Device(MSTR) {
                Name (_PR0, Package{DPR0})
                Name (_PR1, Package{DPR0}) }
            Device(EIO) {
                Device(COM0) {
                    Name(_PRW,…)
                    Method(_PS0) {…}
                    Method(_PS3) {…}
Scope (\_TZ) {
    PowerResource (PFAN, \_S0, 0) {
        Method(_ON)  {…}
        Method(_OFF) {…}}
    Device (FAN) {
        Object (_PR0, Package{PFAN})}
    ThermalZone (THM1) {
        Method(_TMP) {…}
        Method(_SCP, 1) {…}
        Name(_AC1, Package{FAN})
        Name(_PSL, … }}
```

Device Power Management

- Device States (Dx): D0-D3
  - Specific functionality defined in the Device Class Power Management Specifications
- Layered approach
  - Policy owner: decides when to switch states
  - Device driver: saves/restores lost context
  - Bus driver (including ACPI): operates power state controls

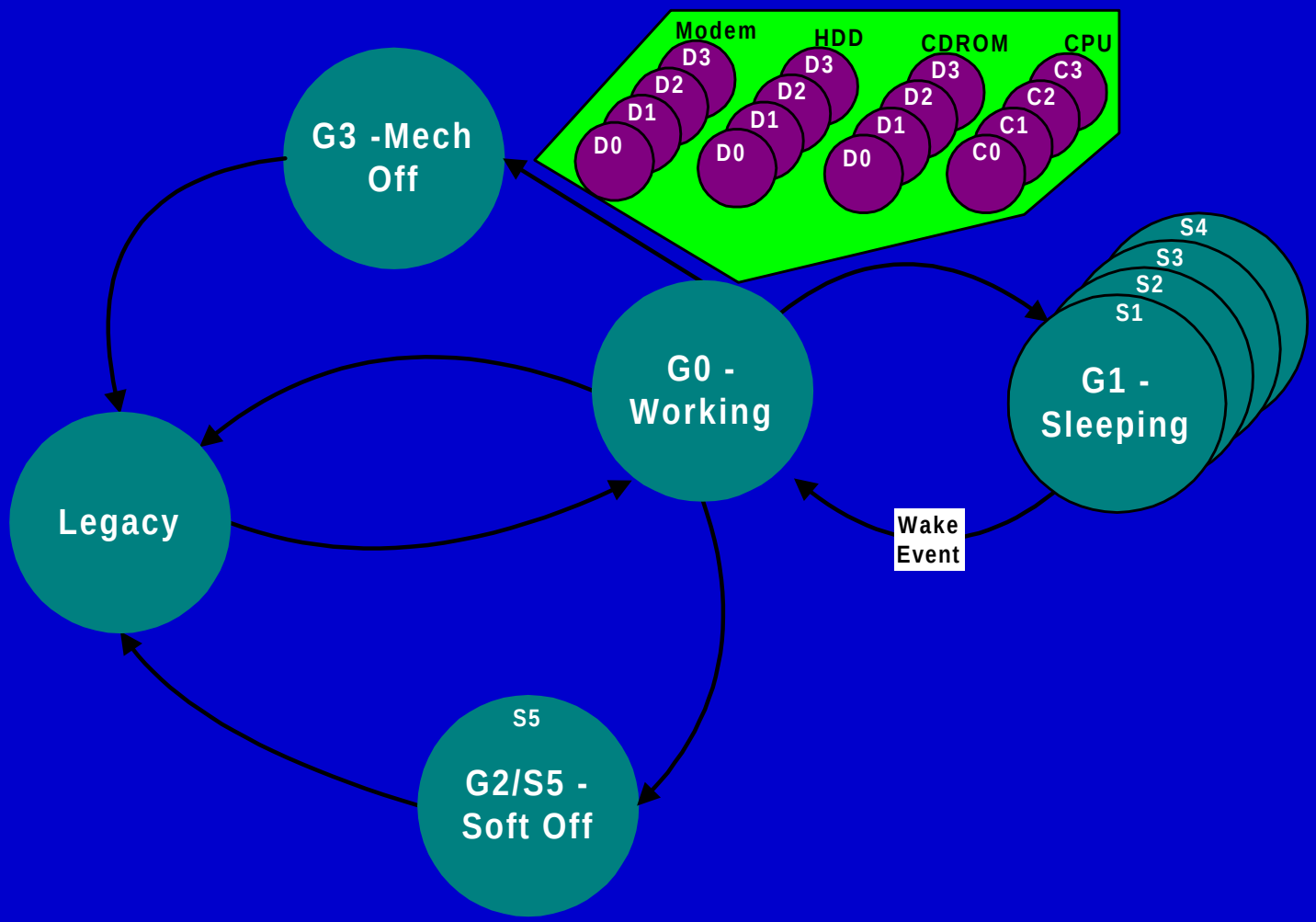
Processor Power Management

- Processor states (Cx); C0-C3
- Based on idle time and bus master activity, OS chooses different Cx states
  - Determined using ACPI PM timer

System Power Management

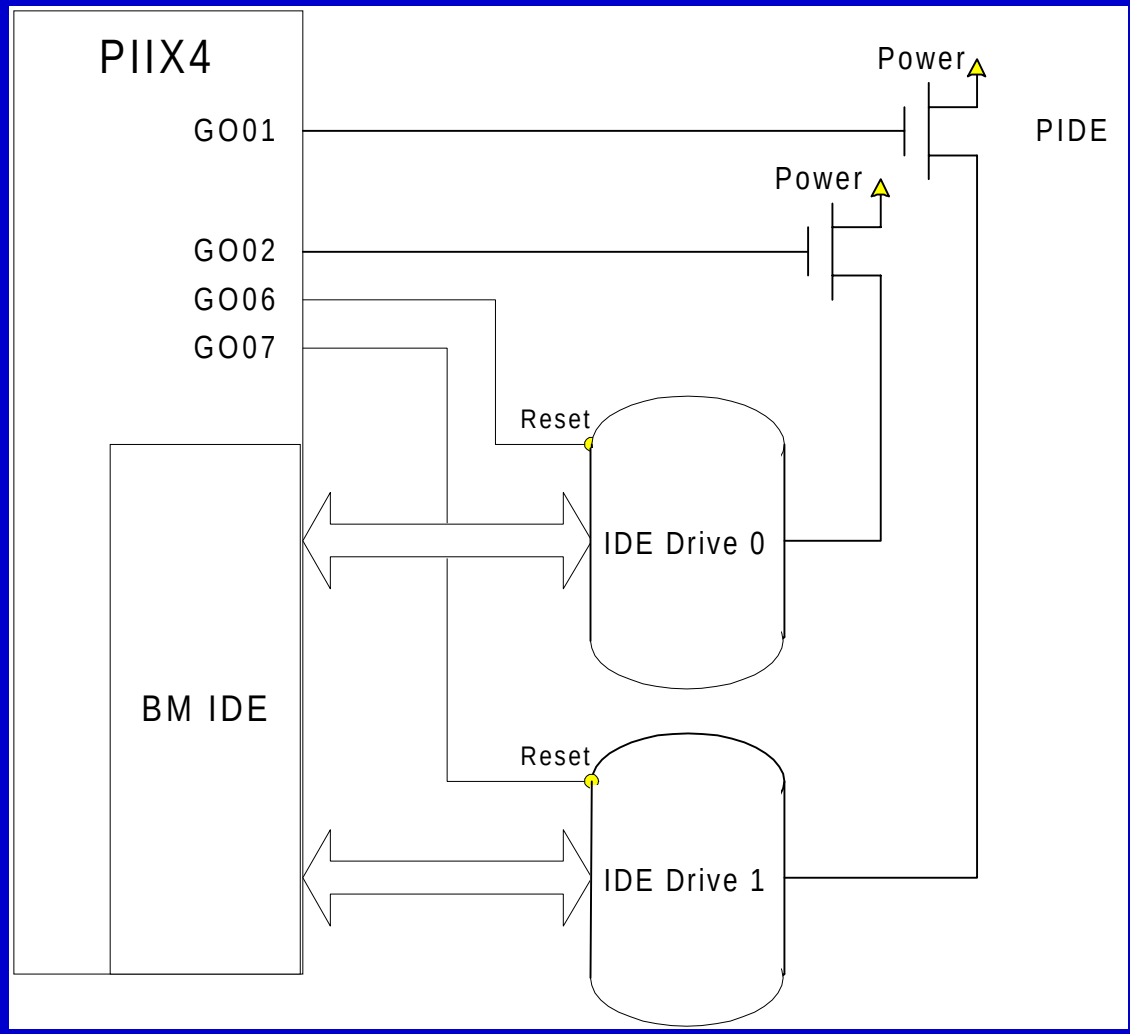
- System states (Sx): S0-S5
  - S0 = fully running
  - S1-S3 = system sleeping; RAM refreshed
  - S4 = hardware off; OS saves memory context
  - S5 = soft-off; no memory context saved

ACPI Power States



Power Management in ACPI

- Device Power Control



Power Management in ACPI

- Device Power Control
  - IDE Power Resource Object

```
PowerResource(PIDE, 0, 0) {
  Method(_STA, 0) {
        Return(Xor(GO01, One, Local0))}
  Method(_ON, 0) {
        Store(Zero, GO01)        // Apply power
        Sleep(10)
        Store(One, GO06)         // Pulse reset
        Stall(10)
        Store(Zero, GO06)}
  Method(_OFF,0) {
        Store(One, GO01)}        // Cut power
}
```

Power Management in ACPI

- CPU Power Control
  - Processor Object

    Processor(_PR.CPU0, 1, 0x1010, 0x06)

    Defines processor control register block at system IO address 1010h - 1015h
  - FACP Table

    DUTY_OFFSET, DUTY_WIDTH fields define processor throttle parameters

Power Management in ACPI

- System Power Control
  - O/S has direct access to h/w registers to initiate system sleep
  - _PTS control method
    - **Called by OS at the beginning of each sleep process**
    - **Save SMM data changed since POST**
    - **Not called just before a suspend!**

Wake up from S1

- From S1
  - BIOS does not get control between sleeping and waking
  - No Context Lost
    - Chipset on
    - CPU on
    - Clocks stopped
    - Memory in suspend refresh
  - After OS is running, it will execute Wake Control Method (_WAK)
    - What to do in _WAK?
      - Notify OS to check for changes in docking station
      - Notify(_SB.PCI0.MPCI, 1)
      - Notify OS to check and set thermal control
      - Notify(_TZ.THRM, 0x81)
      - Notify if h/w added during sleep if no other notify method is used

Wake up from S2

S1 plus CPU context lost (powered off)

- Processor will start executing at power-on reset vector
  - Restore CPU SMRAM base register
  - Pass control to OS
    - Get FACS table pointer through CMOS
    - Retrieve wake vector XYYYZ from FACS
    - Convert physical address XYYYZ to segment:offset format XYYY:Z
    - Start execution at address XYYY:Z
  - After OS is running, it will execute Wake Control Method (_WAK)

Wake Up From S3

Save to RAM, C/S off, DRAM on, CPU off

- Processor will start executing at power-on reset vector
  - Restore Memory Controller
  - Invalidate Cache
  - Restore Chipset Registers with values stored in NVS
  - Restore SMRAM base register
  - Pass Control to OS **(Same as from S2)**
  - After OS is running, it will execute Wake Control Method (_WAK)

Wake Up From S4

Powered off

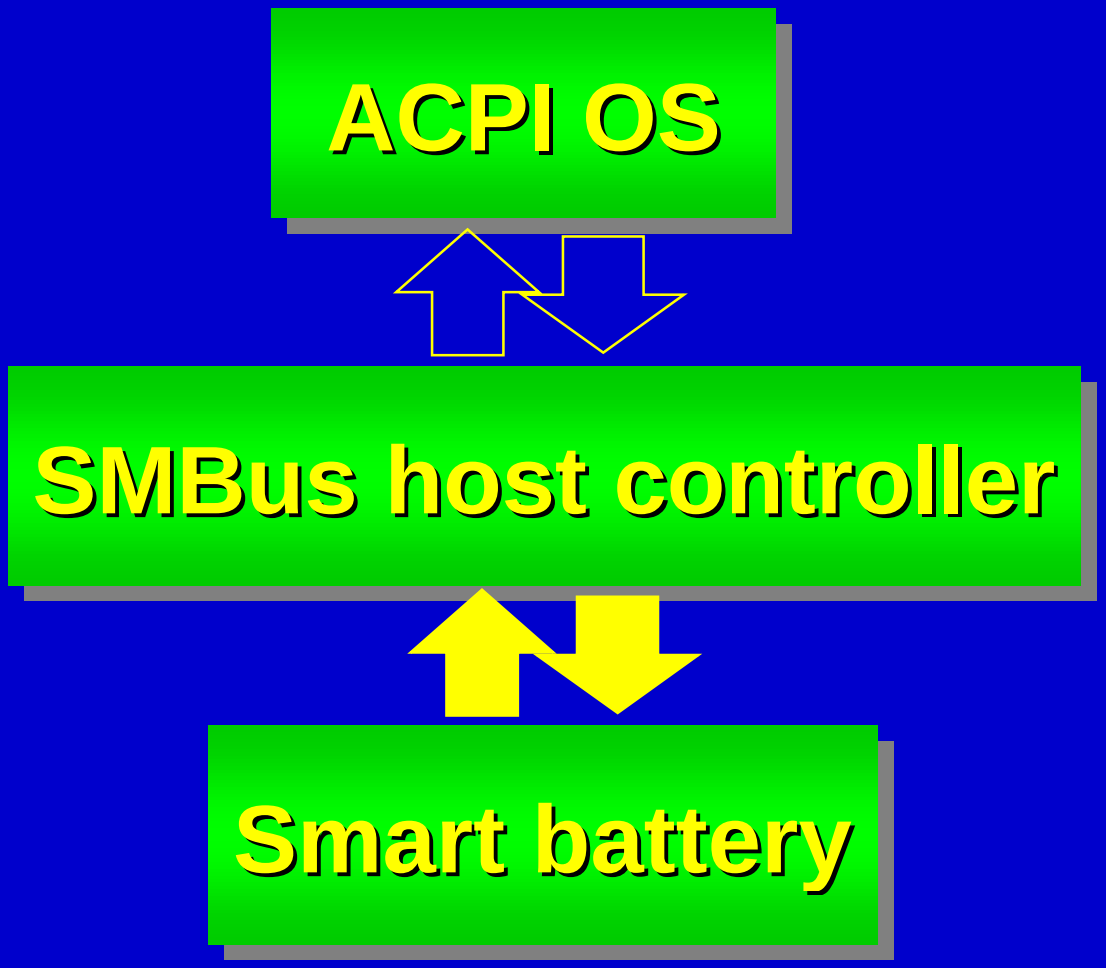
- Two methods
  - S4/OS: OS saves/restores
  - S4/BIOS: BIOS saves/restores
    - Determined by bit in FACP table
- O/S Method
  - Wake up is the same as a cold boot for the BIOS
  - O/S will restore conventional memory
    - O/S compares H/W signature, if no change and other checks pass ...
      - Restores ACPI NVS memory (not atomic)
      - Memory restored prior to calling _WAK
    - If H/W signature changed
      - O/S will not restore ACPI NVS memory
        - I.E. It will cold boot and not restore ANY memory
  - After OS is running, it will execute Wake Control Method (_WAK)

ACPI Battery

- ACPI mobile system has either
  - Smart battery
    - SMBus interface
    - Smart battery driver
  - ACPI control method battery
    - Can use any communication interface
    - OS communicates to the battery via OEM defined control methods

Smart Battery Interface

- Communicate via SMBus interface



SMBus controller

SMBus

Smart battery

Summary

- ACPI OS controls power states of devices, buses and system
- Standard H/W and custom H/W interface is allowed
- Smart Batteries can be accessed with control methods or with OS smart battery driver