Intel Developer Forum
Fall 2000

ACPI 2.0 Specification Technical Review

Guy Therien
ACPI Architecture Mgr.
Mobile Architecture Lab
Intel Corporation
June/July 2000

Intel Labs

Learning Objectives

- Differentiate the changes in ACPI moving from ACPI 1.0b to ACPI 2.0
- Identify the specific ACPI 2.0 changes that support or impact mobile platforms
- Explain how Intel products will be supported using ACPI 2.0 interfaces
- Articulate the time frame for when ACPI 2.0 platform support will be required including the estimated OS support timeline

Agenda

Intel Developer Forum
Fall 2000

- ACPI 2.0 Overview
- Section by Section Review of Changes
- ACPI 2.0 Release Schedule
- Platform Support Timeline
- OS Support Timeline
- Call to Action

Intel Labs

ACPI 2.0 Overview

Intel Developer Forum
Fall 2000

- 64-bit processor / addressing support added
- Processor / device performance states added
- SM Bus CM interfaces rewritten
- Many server related enhancements added
  - hot-pluggable CPUs, memory, GPE Blocks
- Legacy Reduced HW IA-PC support included
- Functional Fixed Hardware concept defined
- General readability/consistency enhancements
- ASL examples updated (corrected)

Intel Labs

Section 1 - Introduction

- Enhanced for readability
- Technical references updated
- Requirements removed!
  - Design guides will now specify required ACPI 2.0 defined interfaces / platform features

Section 2 - Definitions

- Definitions updated, added, removed
  - EFI
  - APIC / SAPIC
  - PSDT removed
- Device and processor performance state definitions added

Section 3 - Overview

Intel Developer Forum Fall 2000

- Battery behavior clarified
  - Batteries must comply with their interface requirements
  - Multi-battery systems are not required to present a composite
  - OS must be notified of changes in battery status
    - Insertion, removal, warning and low levels
- Device and processor performance states added
  - Supports mobile Pentium® III processor featuring Intel® SpeedStep™ Technology
- Thermal management section rewritten

Incorporate Device Performance States In All New Systems

Intel Labs

Section 4 - Hardware

- No hardware changes are required for ACPI 2.0
  - ACPI 1.0 compliant silicon is compliant with ACPI 2.0
- Updated for readability
- Fixed hardware register locations expanded
  - I/O, Memory, PCI Config, Functional Fixed HW
- Processor control moved to section 8
- Server Support
  - GPE block device added

Hardware – continued

- Functional Fixed Hardware
  - CPU manufacturer may provide functional equivalent “fixed hardware” interface comprised of hardware and software
  - May be defined when the interfaces are common across machine designs e.g. systems sharing a common CPU architecture that does not support fixed hardware for all the required interfaces
  - OEMs may only specify interfaces as Functional Fixed Hardware as specified by the CPU manufacturer
  - Intel® SpeedStep™ Technology interfaces employ the Functional Fixed Hardware definition
    - See example later on in this presentation

Hardware - continued

- Reset Register
  - The optional ACPI reset mechanism specifies a standard mechanism that provides a complete system reset. When implemented, this mechanism must reset the entire system. This includes processors, core logic, all buses, and all peripherals. Asserting the reset mechanism is the logical equivalent to power cycling the machine from a software perspective
  - Added to support legacy reduced HW IA-PC

Section 5 - Software

- ACPI system description table definitions contain significant changes
  - Fixed register address space locations expanded
    - New Generic Address Structure allows a register's address space to be specified
  - 64 bit addressing enhancements
  - IA-64 Interrupt controller (SAPIC / IOSAPIC) tables added
  - New fields have been added at end of the system description tables to maintain compatibility with ACPI 1.0

Software - continued

- RSDP Structure extended to allow 64-bit pointer to the new extended RSDT (XSDT)
- Added support for finding the RSDP structure on EFI-enabled systems (IA-64)
- Added XSDT (extended RSDT)
  - Provides identical functionality to the RSDT but accommodates 64-bit physical addresses
  - XSDT supersedes RSDT – ACPI 2.0 OS will look for XSDT first
  - Allows platform to provide one set of tables to an ACPI 1.0 OS and another set of tables to an ACPI 2.0 OS

Software - continued

- FADT Extensions
  - Expanded for 64-bit addressing
  - Preferred PM profile field added
    - System types field used to set default power management policy parameters during OS installation
      - Ease of use enhancement
  - Legacy reduced HW IA-PCs accommodated
    - IA-PC Boot Architecture Flags
      - Set of flags is used by an operating system to guide the assumptions it can make in initializing hardware on IA-PC platforms
    - Added Debug Port Table for Windows* PCs

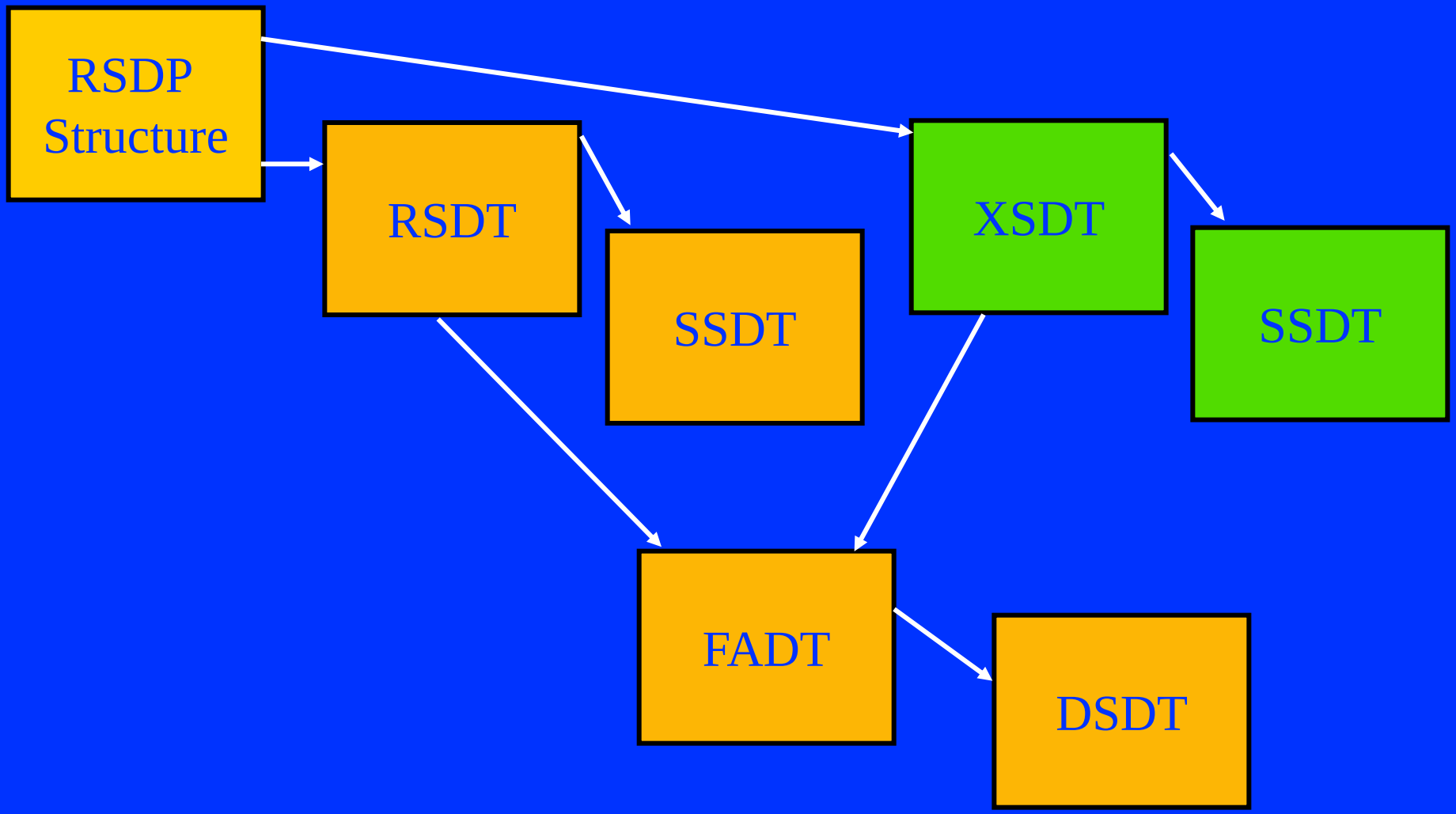
Software - continued

- Added IA-64 Interrupt controller support
  - Added SAPIC / IOSAPIC tables
- New Device Notifications added for
  - Processor, Thermal, and PCI Hot Plug
- Complete list of ALL ACPI-defined generic objects and control methods now provided
- Expanded reserved table signatures
  - DBGP, ECDT, ETDT, HMEM, OEMx
- System type attributes added to fixed feature flags
  - Sealed case, headless
- Removed the PSDT
- Namespace search rules clarified

Software - continued

- _PR and _TZ scopes obsoleted
  - Processors and thermal zones now defined under _SB
- Server Enhancements
  - Introduced GPE blocks implemented via GPE Block device into the event model
- Embedded Controller Boot Resources Table added
  - Enables use of EC operation regions during enumeration
- _REV now evaluates to revision of ACPI that OS implements

ACPI 2.0 System Description Tables



Section 6 - Configuration

- Clarified that an _HID or _ADR is required for each device

- _DMA (DMA Resource)
  - Only defined under devices that represent busses
  - It specifies the ranges the bus controller (bridge) decodes on the child-side of it's interface

- _STR (String)
  - Provides a Unicode string used by the OS to provide information to an end user when describing the device
    - e.g. when device is unknown by OS, this string can be displayed

Configuration - continued

- _INI behavior expanded
- Clarified use of_REG method
  - Make sure you use it!
- Moved floppy related objects to section 10
- Resource Type Specific Flags enhanced
  - Memory types expanded
- All Macros moved to Section 15 – ASL
  - Added Generic Register Descriptor macro

Configuration - continued

- _EDL (Eject Device List)
  - This object evaluates to a package of name space references containing the names of device objects that are dependent on the device under which the _EDL object is declared
  - Allows multiple mobile docks to be correctly represented
- See _EJD / _EDL example on next slide

An example use of _EJD and _EDL with pass-through docks is as follows:

```
Scope(\_SB.PCI0) {

    Device(DOCK1) {     // Pass through dock - DOCK1
        Name(_ADR, …)
        Method(_EJ0, 0) {…}
        Method(_DCK, 1) {…}
        Name(_BDN, …)
        Method(_STA, 0) {0xF}
        Name(_EDL, Package( ) {    // DOCK1 has two dependent devices - IDE2 and CB2
                \_SB.PCI0.IDE2,
                \_SB.PCI0.CB2})
        }
    Device(DOCK2) {     // Pass through dock - DOCK2
        Name(_ADR, …)
        Method(_EJ0, 0) {…}
        Method(_DCK, 1) {…}
        Name(_BDN, …)
        Method(_STA, 0) {0x0}
        Name(_EDL, Package( ) {    // DOCK2 has one dependent device - IDE2
                \_SB.PCI0.IDE2})
        }

    Device(IDE1) {      // IDE Drive1 not dependent on the dock
        Name(_ADR, …)
        }

    Device(IDE2) {      // IDE Drive2
        Name(_ADR, …)
        Name(_EJD," \_SB.PCI0.DOCK1") // Dependent on DOCK1
        }

    Device(CB2) {       // CardBus Controller
        Name(_ADR, …)
        Name(_EJD," \_SB.PCI0.DOCK1") // Dependent on DOCK1
        }
} // end \_SB.PCIO
```

Intel Developer Forum Fall 2000

Intel Labs

Configuration Enhancements for Servers

- _FIX (Fixed Hardware)
  - Provides a correlation between the fixed hardware register blocks and the devices in the ACPI namespace that implement them
- _MAT (Multiple APIC Table Entry)
  - Facilitates hot plugging of APICs
- _PXM (Proximity)
  - Provides topology information conveying proximity of processors and memory enabling CC-NUMA optimizations
- _HPP (Hot Plug Parameters)
  - Specifies the Cache-line size, Latency timer, SERR enable, and PERR enable values for use during hot inserting a PCI device
- _SEG (Segment)
  - Indicates a bus segment location - a level higher than _BBN
    - Each segment has a potential of 256 PCI Bus Numbers

Section 7 – Power and Performance Management

- Added Device Performance State concept

- New _GTS (Going To Sleep) Method
  - Optional. If it exists, OSPM must execute the _GTS control method just prior to setting the sleep enable (SLP_EN) bit in the PM1 control register when entering the S1, S2, S3, and S4 sleeping states and when entering S5 for orderly shutdown

- New _BFS (Back From Sleep) Method
  - Optional. If it exists, OSPM must execute the _BFS method immediately following wake from any sleeping state S1, S2, S3, or S4

Power & Performance - continued

- Clarified _PRx methods must return consistent data
- Clarified _SxD method definitions
- Clarified system sleep state semantics
- Wake events now enabled in S5
  - Enables “Remote Power On”

Section 8 - Processor

- Removed processor power state policy
- Refined processor power state descriptions
- ACPI 2.0 processor objects are declared under the _SB scope
  - Device related objects may appear in the object list
  - A processor driver is implied

New processor object list definitions:

- _PTC (Processor Throttling Control)
  - _PTC - optional object used to define a processor throttling control register alternative to the I/O address spaced-based P_BLK throttling control register (P_CNT)

Processor - continued

- _CST (C States)
  - Optional object that provides an alternative method to declare the supported processor power states (C-States).
  - Supports additional C-States beyond C3
  - Supports dynamic C-states – New notify code defined

- _PCT (Performance Control)
  - Optional object declares an interface that allows OSPM to transition the processor into a performance state
  - _PCT, _PSS and _PPC comprise the ACPI defined interface for controlling Intel® SpeedStep™ Technology performance transitions

Processor - continued

- _PSS (Performance Supported States)
  - Optional object conveys the total number of processor performance states that a system supports

- _PPC (Performance Present Capabilities)
  - Optional object is a method that dynamically indicates the highest number of performance states that is currently supported by the platform

Processor Performance Control ASL Example

- In this example, a uni-processor platform that has processor performance capabilities with support for three performance states as follows:
  - 500-MHz (8.2W) supported at any time
  - 600-MHz (14.9W) supported only when AC powered
  - 650-MHz (21.5W) supported only when docked
- It takes no more than 500 microseconds to transition from one performance state to any other performance state.
- During a performance transition, bus masters are unable to access memory for a maximum of 300 microseconds
- The _PCT - PERF_CTRL and PERF_STATUS registers are implemented as Functional Fixed Hardware
- The following ASL objects are implemented within the system:
  - _SB.DOCK: Evaluates to one if system is docked, zero otherwise.
  - _SB.AC: Evaluates to one if AC is connected, zero otherwise.

Performance Control Example – continued


```
Processor (\_SB.CPU0,              // Processor Name
           1,                      // ACPI Processor number
           0x120,                  // PBlk system IO address
           6 )                     // PBlkLen
      {
      Name(_PCT, Package () // Performance Control object
      { ResourceTemplate() {Register(FFixedHW, 0, 0, 0)},       // PERF_CTRL
        ResourceTemplate() {Register(FFixedHW, 0, 0, 0)}        // PERF_STATUS
      })        // End of _PCT object

      Name (_PSS, Package()
      { Package(){650, 21500, 500, 300, 0x00, 0x08},   // Performance State zero (P0)
        Package(){600, 14900, 500, 300, 0x01, 0x05},   // Performance State one (P1)
        Package(){500, 8200,  500, 300, 0x02, 0x06}    // Performance State two (P2)
      })        // End of _PSS object

      Method (_PPC, 0)          // Performance Present Capabilities method
      {
          If (\_SB.DOCK)
          {
              Return(0)         // All _PSS states available (650, 600, 500).
          }
          If (\_SB.AC)
          {
              Return(1)         // States 1 and 2 available (600, 500).
          }
          Else
          {
              Return(2)         // State 2 available (500)
          }
      }         // End of _PPC method
      }         // End of processor object list
```

The platform issues a *Notify(_SB.CPU0, 0x80)* to inform OSPM to re-evaluate the _PPC object when the number of available processor performance states changes.

Section 9 – Wake / Sleep

- Added _GTS and _BFS control method invocation
- Added requirement for cache flush on entry into S1
  - Driven by Soft Error Rate Q&R guidelines
- Added detailed description of S2 and S3 wakeup
- Added provisions for ACPI only machines
  - Without legacy mode

Section 10 – Device Objects

- Updated _GTF for clarity and added expanded example
- Floppy related objects relocated here
  - _FDI and _FDE
  - _FDM Added - Switches floppy drive mode between 300 and 360 RPM
- Server Enhancements
  - Added GPE Block Device (ACPI0006)
  - Added Module Device (ACPI0004)
    - Resource consumer / server
  - Added Memory Device (PNP0C80)
    - Conveys memory resources in addition to the system address map interfaces
    - Supports hot pluggable memory

Section 11 – Power Source

- Smart Battery Updates
  - Smart Battery System Manager
- Multiple Control Method Battery clarifications and updates
  - Describes how to update temporarily unknown values
  - OSPM requires accuracy so calculated data must be returned rather than hard coded values
  - Clarifications of low, warning, and critical behavior including when to send notifications

Smart Battery-based systems enable the most robust OSPM implementation

Section 12 - Thermal

- _TZD (Thermal Zone Devices)
  - This optional object evaluates to a package of device names. Each name corresponds to a device in the ACPI namespace that is associated with the thermal zone
- Added Notify(*thermal_zone*, 0x82)
  - statement can be used to inform OSPM that a change has been made to the thermal zone device lists (_TZD)
- _TZP (Thermal Zone Polling)
  - This optional object evaluates to a *recommended* polling frequency for a thermal zone
  - Use is strongly discouraged
  - OS is not required to poll at this frequency
- _HOT (HOT - Critical trip point)
  - New trip point where OSPM has enough time to enter S4

Use ACPI-friendly sensors

Section 13 – Embedded Controller

- Multiple clarifications throughout chapter
- Added status code (1F) for CRC return code error
- Added description of packet error checking (PEC - bit 7) of the protocol register
- Added reference to Smart battery system manager spec
- Added multiple SMBus devices to the example ASL code

Section 14 - SMBus

Intel Developer Forum Fall 2000

- New chapter for SMBus
- Includes a completely redesigned SMBus ASL interface
  - Supports all of current SMBus features as well as the ability to operate on non-EC based SMBus segments
  - Defines SMBus Op regions
- Definition of SMBus 1.0 device characteristics under SMBus 2.0 host controllers given to SBS-IF
  - For more info see: http://www.sbs-if.org
- SMBus 2.0 HC added (ACPI0005)

Intel Labs

Section 15 – System Address Map Interfaces

- Old chapter 14

- EFI address map reporting mechanisms added
  - Supports IA-64 systems

Section 16 - ASL

- Old chapter 15
- Added Qword arithmetic support
- Arithmetic constants now available
- Added support for String and Buffer constants
- Break is fixed – now you'll be able use it!
- Added Continue term
- Added Switch term
- Added ElseIF term
- New Operation Regions
  - CMOS
  - PCI BAR Target
- DDB handles and object references can now be passed as parameters to and can be returned from control methods

ASL - continued

- Processor object / thermal zone moved from _PR to _SB
- Added LoadTable function
- Added data table operation region
- Resource macro fields extended
- Useful operators / macros added / enhanced
  - ConcatResTemplate (resource template concatenation)
  - DecStr/HexStr/Int/Buff/String/Copy (explicit data type conversion)
  - Unicode
  - Added Mid operator for strings and buffers
  - Enhanced DerefOf operator to allow string arguments
  - Added Mod operator
  - Index operator operation clarified and now works with strings
- Reserved _T_x for ASL compiler use
- Many syntax clarifications
- SMBus interfaces moved to section 14

Section 17 - AML

- Old Section 16
- Updated for Chapter 16 changes
- Syntax clarifications

Appendix Updates

- Device Class specifications are now Appendix A

- Video Extensions now Appendix B

  New Methods Added:

  - _GPD (Get Post Device)
    - Gets video device to post at next boot (add-in PCI vs. add-in AGP)
  - _SPD (Set Post Device)
    - Sets video device to post at next boot (add-in PCI vs. add-in AGP)
  - _VPO (Video Post Options)
    - Tells OS what video devices are possible to post during boot
    - Motherboard VGA, Add-in PCI VGA, Add-in AGP

ACPI 2.0 Release Schedule

- ACPI 2.0 Is published!
- Download the spec from the teleport site:

  http://www.teleport.com/~acpi

Platform Support Timeline

- PC2001 may be updated to require ACPI 2.0 defined interfaces once ACPI 2.0 is published
  - Required interfaces / platform features will be added
  - Compliance date is unknown
  - We will tell you as soon as we know

- Next System Design Guide will require ACPI 2.0
  - Expected compliance date - June 2002

OS Support Timeline

- Microsoft* will have a phased implementation approach
  - Small subset of interfaces supported in Win64 and BTS 2001 OS releases
  - General support in the OS release after BTS 2001
  - Ask Microsoft for more information

- Linux support will vary with distributor
  - Contact your Linux distributor for more information
  - See also: http://phobos.fachschaften.tu-muenchen.de/acpi/

What we learned today:

Intel Developer Forum Fall 2000

- The changes in ACPI moving from ACPI 1.0b to ACPI 2.0
- The specific ACPI 2.0 changes that support or impact mobile platforms
- How Intel products are supported using ACPI 2.0 defined interfaces
- The probable time frame for when ACPI 2.0 platform support will be required including an estimated OS support timeline

Intel Labs

Call To Action

Intel Developer Forum Fall 2000

- Review the ACPI 2.0 specification
  - http://www.teleport.com/~acpi
- Contact us with implementation questions
  - Use the email reflectors
    - Winpower@hwdev.org (Windows)
    - Acpilist@telelist.com (General)
- Include ACPI 2.0 support in your emerging platforms
  - Ask your BIOS vendor for ACPI 2.0 support
  - Request / implement devices performance state support
- Become an ACPI 2.0 Adopter

Intel Labs

Backup

Thermal Zone Object Requirements

1. All thermal zones must contain the _TMP object.
2. A thermal zone must define at least one trip point -_CRT, _ACx, or _PSV.
3. If _ACx is defined then an associated _ALx must be defined (e.g. defining _AC0 requires _AL0 also be defined).
4. If _PSV is defined then either _PSL or _TZD must be defined. _PSL and _TZD may both be defined.
5. If _PSL is defined then:
   a. If a performance control register is defined (via either P_BLK or _PTC) for a processor defined in _PSL then _TC1, _TC2, and _TSP must be defined.
   b. If a performance control register is not defined (via either P_BLK or _PTC) for a any processor defined in _PSL then the processor must support processor performance states
   (i.e. the processor's processor object must include _PCT, _PSS, and _PPC).
6. If _PSV is defined and _PSL is not defined (i.e. only _TZD is defined) then at least one device in the _TZD device list must support device performance states.
7. _SCP is optional.
8. _TZD is optional outside of the _PSV requirement outlined in #4 above.