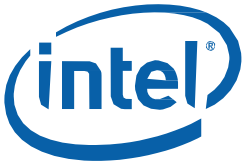

Intel® Processor Vendor-Specific ACPI

Interface Specification

December 2014

Revision 007

Document Number: 302223-007

Contents

Tables

Revision History

Revision Number	Description	Revision Date
001	Initial release	April 2004
002	Updated to include ACPI 3.0 reference information	May 2004
003	FFH description updates	June 2005
004	Expanded _PDC bit definition. Renamed document to comprehend general ACPI Functional Fixed Hardware (FFH) interface definitions and vendor specific object interfaces.	March 2006
005	Added _PDC[11] bit definition	September 2006
006	Added _PDC / _OSC [12] bit definition	January 2014

1 Introduction

1.1 Overview

The Advanced Configuration and Power Interface (ACPI) specification describes a number of interfaces that are vendor specific. These include both the concept of the "Function Fixed Hardware" (FFH) interface, which was introduced in revision 2.0, as well as vendor-specific ACPI namespace object definitions.

For specific interfaces, ACPI allows a functional equivalent interface to be declared and implemented via vendor specific hardware registers along with vendor prescribed software manipulation of these registers. It is the vendor's responsibility to convey the interface's register descriptions and required software manipulation to operating system vendors to gain the correct implementation and support of any feature in a target operating system. This document describes Intel platform specific ACPI FFH and namespace object interfaces.

This document assumes that the reader is familiar with the interfaces defined by the ACPI 3.0 specification. See http://www.acpi.info for more information regarding ACPI 3.0.

§

2 OSPM Capabilities Interfaces

2.1 _PDC (Processor Driver Capabilities)

The optional _PDC (Processor Driver Capabilities) object is a control method that communicates OSPM's processor power management capabilities to the platform. OSPM evaluates the _PDC object before evaluating other objects within the processor object's scope. As such, OSPM's capabilities conveyed by the _PDC object can impact the result of other processor object evaluations. A common example of this is in the platform's exposure of an FFH interface enabling native Processor Performance State transitions via the _PCT object only when OSPM conveys this support capability via _PDC. This _PDC method is evaluated once during processor device initialization, and will not be re-evaluated during resume from a sleep state transition. The platform must preserve state information across S1-S3 sleep state transitions.

OSPM capability bits are defined within the third DWORD of the DWORD argument buffer passed into to the _PDC control method.

Syntax

```
_PDC (PDCBuffer)=> Null
```

Arguments

PDCBuffer

DWordConst Buffer (RevisionID, Count, CapabilitiesDWORD1,…,CapabilitiesDWORDn)

RevisionID

The revision ID of the **CapabilitiesDWORD** format. Current revision is 1.

Count

The number of **CapabilitiesDWORD** values in the buffer.

CapabiltiesDWORD1-n

Capabilities DWords, where each bit defines capabilities and features supported by OSPM for processor configuration and power management as specified by the CPU manufacturer.

Capability coordination between the OSPM and the ACPI firmware will occur as follows:

- OSPM will set appropriate capabilities bits in the _PDC capabilities DWORD
- ACPI firmware will interpret and record the set bits. Return values from additional object evaluations may be impacted by the conveyed OSPM capabilities and platform capabilities.
- OSPM will evaluate additional objects as necessary and act on returned values accordingly.

Table 1. _PDC Capabilities DWORD 2 Bit Definitions

Bit	Definition
0	If set, OSPM is capable of direct access to Performance State MSR's
1	If set, OSPM supports the C1 "I/O then Halt" FFH sequence for multi-processor configurations.
2	If set, OSPM is capable of direct access to On Demand throttling MSR's
3	If set, OSPM is capable of independent C1, P state and T state support for each processor for multi-processor configurations. This bit does not indicate support for asymmetrical _CST, _PSS, or _TSS objects for individual processors in the system.
4	If set, OSPM is capable of independent C2 and C3 state support for each processor for multi-processor configurations. This bit does not indicate support for asymmetrical _CST objects for individual processors in the system.
5	If set, OSPM is capable of native P State software coordination using BIOS supplied _PSD information for multi-processor configurations.
6	If set, OSPM is capable of native C State software coordination using BIOS supplied _CSD information for multi-processor configurations.
7	If set, OSPM is capable of native ACPI throttling software coordination using BIOS supplied _TSD information for multi-processor configurations.
8	If set, OSPM is capable of performing native C State instructions (beyond halt) for the C1 handler in multi-processor configurations.
9	If set, OSPM is capable of performing native C State instructions for the C2/C3 handlers in multi-processor configurations
10	Reserved
11	If set, OSPM is capable of hardware coordination of P States[1]
12	If set, OSPM supports native interrupt handling for Collaborative Processor Performance Control notifications.
13	If set, OSPM is capable of supporting Hardware Duty Cycling.
14-31	Reserved – must be reset to zero.

NOTES:

1. On an IA-32 platform, hardware coordination is done via the ACNT/MCNT feedback mechanism. Please see Appendix B of the *IA-32 Intel® Architecture Software Developer's Manual Volume 3B* for more details.

2.2 _OSC (Operating System Capabilities)

ACPI 3.0 describes a new Operating System Capabilities (_OSC) control method to convey OSPM capabilities to the platform. The functionality of the _OSC object is similar to the _PDC object. As such, the Capabilities DWORD bit definitions defined in Table 1 apply to both the _PDC and the _OSC objects.

Arguments:

```
Arg0 (Buffer):   UUID
Arg1 (Integer):  Revision ID
Arg2 (Integer):  Count
Arg3 (Buffer):   Capabilities Buffer,
```

UUID – Universal Unique Identifier (16-Byte Buffer) used by the platform in conjunction with Revision ID to ascertain the format of the Capabilities buffer. The GUID for Intel® platforms is: **4077A616-290C-47BE-9EBD-D87058713953**.

Revision ID – The revision of the Capabilities Buffer format. Current revision is 1.

Count - Number of DWORDs in the Capabilities Buffer in Arg3

Capabilities Buffer – Buffer containing the number of DWORDs indicated by Count. The first DWORD of this buffer contains standard bit definitions as described below. Subsequent DWORDs contain UUID-specific bits that convey to the platform the capabilities and features supported by OSPM. Successive revisions of the Capabilities Buffer must be backwards compatible with earlier revisions. Bit ordering cannot be changed. For Intel platforms, these bits are defined in Table 1 and are identical to those used by the _PDC object.

The first DWORD in the capabilities buffer is used to return errors defined by _OSC. This DWORD must always be present and may not be redefined/reused by unique interfaces utilizing _OSC.

- Bit 0- Query Support Flag. the _OSC invocation is a query by OSPM to determine which capabilities OSPM may take control of. In this case, _OSC sets bits for those capabilities of which OSPM may take control and clears bits for those capabilities of which OSPM may not take control. If zero, OSPM is attempting to take control of the capabilities corresponding to the bits set.
- Bit 1- Always clear(0).
- Bit 2- Always clear(0).
- Bit 3- Always clear(0).
- All others- reserved.

Result Code:

Capabilities Buffer (Buffer) – The platform acknowledges the Capabilities Buffer by returning a buffer of DWORDs of the same length. Set bits indicate acknowledgement and cleared bits indicate that the platform does not support the capability.

The first DWORD in the capabilities buffer is used to return errors defined by _OSC. This DWORD must always be present and may not be redefined/reused by unique interfaces utilizing _OSC.

- Bit 0- Reserved (not used)
- Bit 1- _OSC failure. Platform Firmware was unable to process the request or query. Capabilities bits may have been masked.
- Bit 2- Unrecognized UUID. This bit is set to indicate that the platform firmware does not recognize the UUID passed in via Arg0. Capabilities bits are preserved.
- Bit 3- Unrecognized Revision. This bit is set to indicate that the platform firmware does not recognize the Revision ID passed in via Arg1. Capabilities bits beyond those comprehended by the firmware will be masked.
- Bit 4- Capabilities Masked. This bit is set to indicate that capabilities bits set by driver software have been cleared by platform firmware.
- All others- reserved.

The use of _PDC is deprecated in ACPI 3.0 in favor of _OSC. For backwards compatibility, _PDC may be implemented using _OSC as follows:

```
Method(_PDC,1)
{
    CreateDwordField (Arg0, 0, REVS)
    CreateDwordField (Arg0, 4, SIZE)

    //
    // Local0 = Number of bytes for Arg0
    //
    Store (SizeOf (Arg0), Local0)

    //
    // Local1 = Number of Capabilities bytes in Arg0
    //
    Store (Subtract (Local0, 8), Local1)

    //
    // TEMP = Temporary field holding Capability DWORDs
    //
    CreateField (Arg0, 64, Multiply (Local1, 8), TEMP)

    //
    // Create the Status (STS0) buffer with the first DWORD = 0
    // This is required to return errors defined by _OSC.
    //
    Name (STS0, Buffer () {0x00, 0x00, 0x00, 0x00})

    //
    // Concatenate the _PDC capabilities bytes to the STS0 Buffer
    // and store them in a local variable for calling OSC
    //
    Concatenate (STS0, TEMP, Local2)

    // Note: The UUID passed into _OSC is CPU vendor specific.
    // Consult CPU vendor documentation for UUID and Capabilities Buffer
    // bit definitions
    //
    _OSC (ToUUID("4077A616-290C-47BE-9EBD-D87058713953"), REVS, SIZE, Local2)
}
```

§

3 Functional Fixed Hardware Interfaces

3.1 Functional Fixed Hardware

This section defines the Functional Fixed Hardware (FFH) interfaces that are specific to Intel platforms. The ACPI–defined Generic Address Structure (GAS) is used to convey FFH information to OSPM. This 12-byte structure normally describes fixed hardware registers. By encoding the special FFH identifier (0x7F) in the Address_Space_ID field, other structure entries may take on new meanings as described by this document. When the FFH identifier is specified in the Address_Space_ID field and the other GAS fields contain zero values, the platform assumes and requires that OSPM has implicit knowledge of the register mechanisms and manipulations needed to achieve the interface purpose. The table below describes Intel's definitions of non-all-zero field encodings for GAS structure fields when FFH is employed:

Table 2. Intel FFH GAS Structure Field Names

Field	Byte Length	Byte Offset	Intel Field Name
Address_Space_ID	Fixed Functional Hardware (0x7F)	0	Functional Fixed Hardware (GAS type)
Register_Bit_Width	1	1	Vendor Code: Intel = 0x01
Register_Bit_Offset	1	2	Class Code
Access Size	1	3	Arg1 (8-bits)
Address	8	4	Arg0 (64-bits)

The contextual usage of FFH is associated with a specific use in the namespace, for example, it may imply a specific register sequence for a particular C state or P state transition. The following sections describe ACPI Object interfaces and corresponding FFH GAS structure encoding of Intel fields.

3.2 _PCT (Performance Control)

_PCT object FFH usage is reserved for Enhanced Intel SpeedStep® Technology. The generic register set defines an extended CPUID feature flag that indicates Enhanced Intel SpeedStep Technology capability. This capability implies the presence of the IA32_PERF_CTL and IA32_PERF_STS registers. When FFH is used in the _PCT object definition, the platform assumes that OSPM has implicit knowledge of the location and pertinent bit fields of the performance states control registers needed to perform performance state transitions.

Table 3. _PCT FFH GAS Field Encoding

Description	Namespace Location	Type	Vendor Code	Class Code (Bit Offset)	Arg0 (Address)	Arg1 (Access Size)
Enhanced Intel SpeedStep® Technology MSR	_PCT	0x7F	0x00	0	0	0

3.3 _CST (C States)

The C1 halt sequence is the standard IA32 processor sequence for placing a given processor into the halt state. C1 entry is performed by issuing a ***STI*** instruction (enable interrupts), followed by the ***HLT*** instruction. This sequence is unchanged from current OS behavior as defined by ACPI.

The C1 I/O then Halt encoding includes a modified C1 entry sequence for use on operating systems that support firmware based C state coordination. This sequence maintains the existing IA32 C1 entry sequence, but precedes the sequence with an I/O read to the specified I/O port address. The width of the I/O read is required to be a byte-wide (8 bit) access.

The Native C State encoding supports processors that support the MWAIT instruction and MWAIT Extensions if applicable. MWAIT Hints correspond to the value passed via the EAX register when executing the MWAIT instruction.

Table 4. _CST FFH GAS Field Encoding

Description	Namespace Location	Type	Vendor Code	Class Code (Bit Offset)	Arg0 (Address)	Arg1 (Access Size)
C1 Halt	_CST	0x7F	0x00	0	0	0
C1 I/O then Halt	_CST	0x7F	0x01	1	I/O port address (byte access)	0
Native C State Instruction (Beyond halt)	_CST	0x7F	0x01	2	[63:32] Reserved [31:00] Hints	Bit 0: Set to 1 if HW-coordinated Bit 1: Set to 1 if OSPM should use Bus Master avoidance for this C-state

3.4 _PTC (Processor Throttling Control)

_PTC object FFH usage is reserved for clock modulation. This capability implies the presence of clock modulation specific registers. When FFH is used in _PTC, it is assumed that OSPM has implicit knowledge of the location and pertinent bit fields of the throttling state control registers needed to perform throttling state transitions.

Table 5. _PTC FFH GAS Field Encoding

Description	Namespace Location	Type	Vendor Code	Class Code (Bit Offset)	Arg0 (Address)	Arg1 (Access Size)
Clock Modulation MSR	_PTC	0x7F	0x00	0	0	0

§