INTEL LOW POWER S0 IDLE

May 2017

Revision 002

Contents

Revision History

Revision Number	Description	Revision Date
001	Initial release	July 2014
002	Added Low Power Idle _DSM Definitition	May 2017

1. Introduction

ACPI 5.0 introduced the concept of a Low Power S0 Idle Capable platform through setting FADT.Flags[21] bit. The setting of this bit infers the presence of one or more Low Power Idle (LPI) states on such an LPI capable platform.

The inference of LPI state(s) is insufficient for a contemporary OS to monitor and provide meaningful diagnostics of whether the LPI state was entered, for how long and diagnose if the desired state was not achieved.

Intel is using a defined table to provide this information. The table has a reserved signature ("LPIT") in the ACPI specification, and must be included in the set of ACPI tables provided by Intel LPI capable platforms.

Intel is also using a set of Device Specific Methods (_DSM) that can be used by Operating System Software to interact with system firmware to optimize power and functionality around transitions into and out of the Low Power S0 Idle state(s).

This document assumes that the reader is familiar with interfaces defined by the ACPI specification. See http://www.uefi.org/specifications for information regarding the latest ACPI specification.

2. Low Power Idle Table (LPIT) definition

To enumerate platform Low Power Idle states, the system will use the "Low Power Idle Table" (LPIT). The LPIT consists of a standard ACPI header followed by a series of one of more LPI State descriptors.

2.1. LPIT Table Main Structure

Table 1 Structure ACPI Table Header

Field	Byte Length	Byte Offset	Value	Description
ACPI Header				
Signature	4	0	"LPIT"	Signature for the table.
Length	4	4	36 + <sum of all LPI State Structures>	Length, in bytes, of the entire table
Revision	1	8	0	Revision
Checksum	1	9	<checksum>	Entire table must sum to zero.

OEMID	6	10	<firmware-specific>	OEM ID
OEM Table ID	8	16	<firmware-specific>	The table ID is the manufacturer model ID.
OEM Revision	4	24	<firmware-specific>	OEM revision for supplied OEM Table ID.
Creator ID	4	28	<firmware-specific>	Vendor ID of utility that created the table.
Creator Revision	4	32	<firmware-specific>	Revision of utility that created the table.
LPI State Structure[n]	varies	36	<implementation-specific>	See Section 2.2

2.2. LPI State Descriptor

LPI state descriptor provides OSPM with additional characteristics including entry trigger, residency & latency requirements and associated residency counter descriptor. If multiple LPI states exist, the more shallow LPI states are expected to have smaller residency & latency requirements (and higher power). If multiple LPI states are defined, they must be ordered from shallowest to deepest with a zero-based monotonically increasing value (0..n) Unique ID. Multiple LPI state descriptors may exist for a single Unique ID, but only one may be enabled (as indicated in Flags) per Unique ID.

Table 2 Low Power Idle Structure Types

Value	Description
0	Native C-state instruction based LPI structure type
> 0	Reserved for future use

LPI Structure Types > 0 are reserved for future use. If additional LPI Structure Types are added, this document will be updated to reflect those changes.

2.2.1. Native C-state instruction based LPI structure type

LPI structure with Native C-state instruction entry trigger descriptor.

Table 3 LPI Structure with MWAIT Entry Trigger Descriptor

Field	Byte Length	Byte Offset	Description
Type	4	0	LPI State Descriptor Type 0
Length	4	4	Length of LPI State Descriptor Structure
Unique ID	2	8	Unique LPI state identifier: zero based, monotonically increasing identifier
Reserved	2	10	Must be zero
Flags	4	12	See Flags descriptor in Table 4
Entry Trigger	12	16	The LPI entry trigger, matching an existing _CST. Register object, represented as a Generic Address Structure. All processors must request this state or deeper to trigger.
Residency	4	28	Minimum residency or “break-even” in microseconds (uS)
Latency	4	32	Worst case exit latency in microseconds uS
Residency Counter	12	36	[optional] Residency counter, represented as a Generic Address Structure. If not present, Flags[1] bit should be set.
Residency Counter Frequency	8	48	Residency counter frequency in cycles per second. A value of 0 indicates that counter runs at TSC frequency. Valid only if Residency Counter is present.

For optional registers that are not supported by the platform, the following NULL register descriptor should be used to indicate this to the host:

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

2.2.1.1. Flags Field

Flags field is generic to all LPI structure types

Table 4 Flags Field

Flag Field	Bit Length	Bit Offset	Description
Disabled	1	0	If set, LPI state is not used

Counter Not Available	1	1	If set, Residency counter is not available for this LPI state and Residency Counter Frequency is invalid
Reserved	30	2	Reserved for future use. Must be zero

2.2.1.2. Fixed Functional Hardware-based Residency Counter (MSR)

If the LPI residency counter is a Model Specific Register (MSR), it cannot be natively described in an ACPI Generic Address Structure (GAS). Therefore it must be described as Fixed Functional Hardware (FFH) with the GAS field requirements:

Table 5 GAS Field Requirements for Fixed Functional Hardware Device

Field	Byte Length	Byte Offset	Value	Description
Address Space ID	1	0	0x7F	Fixed Functional Hardware
Register Bit Width	1	1	64	64 to indicate the MSR bit width
Register Bit Offset	1	2	0	Must be zero
Access Size	1	3	0	Must be zero
Address	8	4	<Implementation-specific>	MSR value

3. Low Power S0 Idle Device-Specific Method (_DSM) Definition

In order for the OS to interact with the Intel platform, an ACPI Device must be exposed through the Namespace. This Device must include a _CID object containing EISAID(“PNP0D80”). The following _DSM definition must be contained within the scope of this Device.

Table 6 _DSM Definition for INT3381

UUID	Revision	Function	Description
c4eb40a0-6cd2-11e2-bcfd-0800200c9a66	0	0	Enum Functions
	0	1	Get Device Constraints
	0	2	Get Crash Dump Device
	0	3	Display Off Notification
	0	4	Display On Notification
	0	5	Low Power S0 Entry Notification
	0	6	Low Power S0 Exit Notification

3.1. Get Device Constraints (Function 1)

For every Intel SoC, there is a set of hardware preconditions that must be met for the SoC to achieve its lowest power platform idle state. These hardware preconditions or "constraints" are generally related to individual device power states (e.g. D3). The hardware constraints are translated into equivalent software states that can be tracked by OS Power Management. For each platform device in a Low Power S0 Idle system, the Platform Idle State constraints are specified in terms of minimum D state or Device specific state. The constraint is satisfied if the device enters the target state or a deeper device state. In other words, if a Device specific state is described as a constraint, the constraint is met if the device transitions to either the described Device-specific-state or deeper D-state. The Device enabled/disabled parameter enables BIOS to enumerate all possible devices and dynamically enable/disable as needed during BIOS initialization.

Arguments:

Arg0: UUID: c4eb40a0-6cd2-11e2-bcfd-0800200c9a66
Arg1: Revision ID: 0
Arg2: Function Index: 1
Arg3: Unused

Return:

A Package of device constraint entries, described in Table 7.

Table 7 Device Constraint Entry Format

Field Name	Format	Descritption
Device Name	String	Fully qualified Namestring
Device Enabled	Integer	0=Disabled, no constraints Non-zero=Device is enabled and constraints apply
Device Constraint Detail	Package	see Table 8

Table 8 Revision Package Format

Field Name	Format	Description
Revision	Integer	Integer (zero)
Constraint Package	Constraint Package	Package (see Table 9)

Table 9 Constraint Package

Field Name	Format	Description
LPI UID	Integer	LPIT entry (see Section 2.2) 0xFF: Constraint applies to all entries in LPIT
Minimum D-state Precondition	Integer	Minimum D-state constraint. 0 = D0 1 = D1 2 = D2 3 = D3
Minimum device-specific state Precondition	Integer	Optional device-specific information

Example encoding of the package returned for Function 1:

```
Package()
{
  Package() {"\\_PR.CPU0", 0x1, Package() {0, Package(){0xFF, 0}}},
  Package() {"\\_PR.CPU1", 0x1, Package() {0, Package(){0xFF, 0}}},
  Package() {"\\_SB.PCI0.GFX0", 0x1, Package() {0, Package(){0xFF, 3}}},
  Package() {"\\_SB.PCI0.SAT0", 0x1, Package() {0, Package(){0xFF, 0, 0x81}}}
}
```

3.2. Get Crash Dump Device (Function 2)

This function supplies operating system software with a simple mechanism to ensure that a listed storage device can be brought into a state where a crash dump file can be stored. All potential crash dump storage devices must be listed in the package returned by this function

Arguments:

Arg0: UUID: c4eb40a0-6cd2-11e2-bcfd-0800200c9a66

Arg1: Revision ID: 0
Arg2: Function Index: 2
Arg3: Unused

Return:

A Package, as described in Table 10.

Table 10 Crash Dump Device Package

Field Name	Format	Descritption
DeviceName	String	Fully qualified Namestring
PowerControl	Package	List of one or more operations to bring the Device into an operational state. See Table 11

Table 11 Power Control

Field Name	Format	Descritption
Register	Package	GAS (as defined in the ACPI spec)
Operation	Package	See Table 12

Table 12 Power Control Operation

Field Name	Format	Descritption
ANDMask	Integer	Used for Trigger types 3 and 4
Value	Integer	Value written for Trigger types 1-4
Trigger	Integer	0 = Read 1 = Write 2 = Write followed by Read 3 = Read Modify Write 4 = Read Modify Write Read Other values are reserved
Delay	Integer	Time, in microseconds, to delay after completion of this operation.

The “AND mask” and “Value” package components must match the “Register Bit Width” as described in the associated GAS structure package. The “AND mask” is only applicable to “Read Modify” triggers 3 (Read Modify Write) and 4 (Read Modify Write followed by Read). For these triggers, the register is read, ANDed with the Mask, ORed with the Value, and written back. For type 4, the register is read again to ensure that the data written has been flushed to the hardware component containing the register.

3.3. Display Off Notification (Function 3)

This _DSM Function may be invoked at such point in time that the operating system has entered a state where all displays (local and remote, if any) have been turned off. This could occur based on some user action, e.g. a button press or lid close event, or expiration of some display power down timer. If OSPM supports Display Off Notifications by invocation of this function, it must also invoke the Display On Notifications described in the next section.

Arguments:

Arg0: UUID: c4eb40a0-6cd2-11e2-bcfd-0800200c9a66
Arg1: Revision ID: 0
Arg2: Function Index: 3
Arg3: Unused

Return:

None

3.4. Display On Notification (Function 4)

This _DSM Function must be invoked if a Display Off Notification has occurred and any display (local or remote) is returned to an active state. If OSPM supports Display On Notifications by invocation of this function, it must also invoke the Display Off Notifications described in the previous section.

Arguments:

Arg0: UUID: c4eb40a0-6cd2-11e2-bcfd-0800200c9a66
Arg1: Revision ID: 0
Arg2: Function Index: 4
Arg3: Unused

Return:

None

3.5. Low Power S0 Entry Notification (Function 5)

This _DSM function may be invoked when the OS state has reached sufficient criteria for processor idle state entry matching Entry Trigger defined in LPIT to be interpreted as a request for the platform to enter a Low Power S0 Idle (LPI) state. If OSPM supports Low Power S0 Entry Notifications by invocation of this function, it must also invoke the Low Power S0 Exit Notifications described in the next section.

Arguments:

Arg0: UUID: c4eb40a0-6cd2-11e2-bcfd-0800200c9a66
Arg1: Revision ID: 0
Arg2: Function Index: 5
Arg3: Unused

Return:

None

3.6. Low Power S0 Exit Notification (Function 6)

This _DSM function may be invoked when the OS state is no longer sufficient for processor idle state entry matching Entry Trigger defined in LPIT to be interpreted as a request for the platform to enter a Low Power S0 Idle (LPI) state. If OSPM supports Low Power S0 Exit Notifications by invocation of this function, it must also invoke the Low Power S0 Entry Notifications described in the previous section.

Arguments:

Arg0: UUID: c4eb40a0-6cd2-11e2-bcfd-0800200c9a66
Arg1: Revision ID: 0
Arg2: Function Index: 6
Arg3: Unused

Return:

None