ACPI LOW POWER IDLE TABLE (LPIT)

July 2014

Revision 001

ACPI LOW POWER IDLE TABLE (LPIT)

Contents

Revision History

Revision Number	Description	Revision Date
001	Initial release	July 2014

1. Introduction

ACPI 5.0 introduces 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 RSDT of Intel Windows 8.1 and later LPI capable platforms that support Microsoft's inbox *Intel(R) Power Engine Plug-in* driver.

This document assumes that the reader is familiar with interfaces defined by the ACPI specification. See http://www.uefi.org 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. ACPI Table Header

Table 1 - 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>	

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.

LPI Structure Types

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 - Native C-state instruction based LPI structure

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
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 uSec
Latency	4	32	Worst case exit latency in uSec
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	[optional] Residency counter frequency in cycles per second. Value 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

ACPI LOW POWER IDLE TABLE (LPIT)

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 - Fixed Functional Hardware-based Residency Counter (MSR)

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