# **P**rocessor **C**locking **C**ontrol (PCC) Interface Specification

Version 1.0
February 23, 2009
Hewlett-Packard and Microsoft Corporation

# NOTICE

The information contained in this document is subject to change without notice.

Both Hewlett-Packard and Microsoft irrevocably promise, with respect to the PCC Specification, not to assert any of their respective Necessary Claims against any Implementers of the PCC Specification for making, using, selling, offering for sale, importing or distributing any implementation of the PCC Specification to the extent such implementation conforms to the PCC Specification, and is compliant with all of the required parts of the mandatory provisions of the PCC Specification ("Covered Implementation"), except where the Implementer asserts a claim against, or files, maintains or voluntarily participates in a lawsuit, arbitration, administrative action or other judicial or quasi-judicial proceeding against a Microsoft or Hewlett-Packard Covered Implementation. Here, "Implementer" means any party to this promise that implements a Covered Implementation. "Necessary Claims" are those claims of (i) Microsoft-owned or Microsoft-controlled patents, (ii) Hewlett-Packard-owned or Hewlett-Packard-controlled patents, or (iii) patents jointly owned or controlled by Microsoft and Hewlett-Packard, that are necessary to implement the required portions (which also include the required elements of optional portions) of the Specification that are described in detail and not merely referenced in the Specification. A patent is "controlled" by a party if (a) the patent is owned by an Affiliate of that party or (b) that party or its Affiliate, is the exclusive licensee of the patent; and that party, or its Affiliate, has the right to make this promise regarding the patent without a duty or obligation to pay a royalty or other fee to a third party. "Affiliate" means, in relation to a party, any entity controlled, directly or indirectly, by that party, any entity that controls, directly or indirectly, that party or any entity directly or indirectly under common control with that party.

This promise is not an assurance either (i) that any of Microsoft's or Hewlett-Packard's issued patent claims covers a Covered Implementation or are enforceable or (ii) that a Covered Implementation will not infringe the patents or other intellectual property rights of any third party. No other rights except those expressly stated in this promise shall be deemed granted, waived or received by implication, exhaustion, estoppel or otherwise."

# Contributors



| Name | Email |
|---|---|
| **Scott Faasse** | **scott.faasse@hp.com** |
| **Nick Judge** | **nicjudge@windows.microsoft.com** |
| **Tristan Brown** | **tristanb@microsoft.com** |
| **Stephen Berard** | **sberard@microsoft.com** |

# 1. Introduction

A processor's performance is typically proportional to the speed of its internal clock frequency. Many processor vendors provide mechanisms for software control of the clock frequency. These controls are used by power management software in order to reduce power when maximum performance is not required.

Many issues arise when more than one software or firmware entity wants to control these mechanisms. Typically, the platform firmware ultimately decides whether or not the control hardware is abstracted to the Operating System. Existing ACPI methods for abstraction do not allow for the power management features provided by the platform firmware and the Operating System to co-exist. This specification details an alternative mechanism for coordinating processor clocking control.

The Processor Clocking Control (PCC) interface is implemented by platform firmware in order to provide a channel for the Operating System to direct and obtain performance information on a per-processor basis in cases where the platform firmware would normally be directing the control of the processor performance. In this model the platform firmware remains in direct control of the processor clocking control registers. The Operating System computes the required performance for each processor and communicates this to the platform firmware via the PCC shared memory interface. The platform firmware is responsible for managing the hardware clocking controls in order to deliver the requested performance. The interface also provides the capability for the Operating System to obtain the actual performance level delivered. In cases where the platform is unable to meet the Operating System request, such as a thermal or power budget conditions, flags are set to indicate this to the Operating System.

The PCC interface consists of the following key elements:

- Reserved region of system memory that has shared access between the Operating System and the platform firmware.
- Command/status structure is used along with input and output buffers to communicate status and control
- Alerting mechanism for the Operating System to signal to the platform that a command has been requested (commonly referred to as a "doorbell")
- An optional method for alerting the Operating System that a command has been completed.
- ACPI methods to describe the location of the memory region and the platform doorbell.

# 2. Shared Memory Region

The Processor Clocking Control interface relies on a reserved area in the system memory map for communications between the platform firmware and the Operating System. This region of memory has the following requirements:

- MAY be cached or uncached system memory
- MAY be actual DRAM memory or memory mapped I/O
- MUST NOT be direct I/O access
- MUST be 4KB aligned
- MUST reside in a reserved region as returned by INT 15 E820h or EFI system memory map calls
- MUST be a contiguous region.

The shared memory region is a single header structure in memory. The region of memory is specified in the SharedMemoryRegion field obtained by evaluating the PCCH() method. The region contains a header followed by one input and one output buffer for each logical processor.

## 2.1 Header

The location of the header is at the beginning of the Shared memory region. It is defined as follows:

| Offset | Length | Description |
|---|---|---|
| 0x00 | 4 Bytes | Signature |
| 0x04 | 2 Bytes | Header Length |
| 0x06 | 1 Byte | Major Version |
| 0x07 | 1 Byte | Minor Version |
| 0x08 | 4 Bytes | Supported Features |
| 0x0C | 2 Bytes | Command Field |
| 0x0E | 2 Bytes | Status Field |
| 0x10 | 4 Bytes | Nominal Latency |
| 0x14 | 4 Bytes | Minimum Time Between Commands |
| 0x18 | 4 Bytes | Maximum Time Between Commands |
| 0x1C | 4 Bytes | Nominal CPU Frequency |
| 0x20 | 4 Bytes | Minimum CPU Frequency |
| 0x24 | 4 Bytes | Minimum CPU Frequency w/o Throttling |

### 2.1.1 Signature

This 32-bit field contains a signature that when read as a DWORD will spell out "$PCC" in ASCII text (i.e. "$PCC" = 0x24504343). The signature is used to determine if the memory region is valid. The signature can be used to determine if the memory region is valid. The platform firmware is responsible for populating this field.

## 2.1.2 Header Length

This 16-bit field contains the total length of the variable-length header. The granularity is in bytes. The platform firmware is responsible for populating this field.

## 2.1.3 Major/Minor Version

These two 8-bit fields contain a Major and Minor version of the specification used to define the remaining fields. The values are binary-coded decimal. An implementation of the interface coinciding to version 1.0 of this specification would have a Major revision of 0x01 and a Minor revision of 0x00. The platform firmware is responsible for populating this field.

## 2.1.4 Supported Features

This 32-bit field contains the supported features provided by the platform. This field is a bitwise field where each bit indicates support for a specific feature or capability. The platform firmware is responsible for populating this field.

| Bit | Description | Value |
|---|---|---|
| 0 | SCI doorbell | 1 = Platform is capable of generating a generic SCI to indicate completion of a command.<br>0 = Not supported |
| 31:1 | Reserved | Must be zero |

## 2.1.5 Command Field

This 16-bit field is used to select one of the define commands for the platform to perform. For a list of commands, please refer to Section 3 "Commands." The Operating System is responsible for populating this field.

The command field follows the following format:

| Bit | Description | Value |
|---|---|---|
| 7:0 | Command | Command Code (see section 3 for list of defined commands.) |
| 14:8 | Reserved | Must be zero |
| 15 | SCI doorbell | 1 = Platform generates an SCI upon command completion.<br>0 = Unsupported<br><br>If the SCI doorbell bit in the *Support Features* field indicates that the platform does not support SCI generation. The Operating System must set this bit to 0. |

## 2.1.6 Status Field

This 16-bit field contains status information for the PCC interface. This field is a bitwise field where each bit conveys specific status information. The platform firmware is responsible for populating this field.

| Bit | Description | Value |
|---|---|---|
| 0 | Command Complete | 1 = Platform has completed processing the last issued command. *See Note 1 |
| 1 | SCI doorbell | 1 = Platform generated an SCI. *See Note 2 |
| 2 | Error | 1 = An error occurred executing the last command |
| 15:3 | Reserved | Must be zero |

Note:

1. The Operating System (either in an SCI handler or via polling) is required to detect that the Command Complete bit has been set and clear it before issuing another command. If waiting for this bit to be set, the Operating System must not modify the command or any processor input fields.
2. The SCI doorbell bit is required to be cleared in the Operating System's SCI handler so that a new event can be detected.

## 2.1.7 Nominal Latency

This 32-bit field contains the expected latency for which the platform can complete any command supplied. Time is expressed in units of microseconds. The platform firmware is responsible for populating this field.

## 2.1.8 Minimum Time Between Commands

This 32-bit field contains the amount of time that should expire, expressed in microseconds, which the platform recommends between issuing the same command. It is expected that the Operating System will NOT issue the same command at an interval quicker than this value. The platform firmware is responsible for populating this field.

### 2.1.9 Maximum Time Between Commands

This 32-bit field contains the maximum amount of time, expressed in microseconds, between issuing commands that require the platform to return frequency status. If this timing requirement is not met by the Operating System, certain commands that retrieve frequency status may temporarily return inaccurate results. The platform firmware is responsible for populating this field.

### 2.1.10 Nominal CPU Frequency

This 32-bit field contains the nominal frequency of the logical processor in units of megahertz. The nominal frequency is defined as the maximum frequency of the processor if no clock throttling, over clocking, or clock stretching occurs. The platform firmware is responsible for populating this field.

### 2.1.11 Minimum CPU Frequency

This 32-bit field contains the minimum frequency that the processor could achieve if the clock throttling and stretching controls are used. Frequency is reported in units of megahertz. The platform firmware is responsible for populating this field.

### 2.1.12 Minimum CPU Frequency w/o Throttling

This 32-bit field contains the minimum frequency that the processor could achieve if the clock stretching controls are used but clock throttling controls are unused. Frequency is reported in units of megahertz. The platform firmware is responsible for populating this field.

## 2.2 Processor Input Buffer

Each logical processor has a 4-byte input buffer. The buffer is located at a unique address in the shared memory region. The offset to the input buffer is specified by each processor's PCCP method (described in Section 5.3.) The bit-wise description of the

input buffer is command specific. The platform is expected to initialize the input buffer of each processor to zero. The Operating System is responsible for populating the input buffer.

## 2.3 Processor Output Buffer

Each logical processor has a 4-byte output buffer. The buffer is located at a unique address in the shared memory region. The offset to the output buffer is specified by each processor's PCCP method (described in Section 5.3.) The bit-wise description of the output buffer is command specific. The platform firmware is responsible for populating the output buffer.

# 3. Commands

The shared memory region header contains the command and status interface. The following sections define the command, status, input buffer, and output buffer fields for each command.

## 3.1 Get Average Frequency

The Get Average Frequency command is used by the Operating System to query the running frequency of the processor since the last time this command was last completed. The command returns the running frequency as a ratio of the nominal CPU frequency.

Every logical processor's input and output buffers are evaluated during this command.

| **Input** | | **Value** |
|---|---|---|
| Command Field | Bits 14:0 | X000:0000:0000:0000 |
| | Bit 15 | Set if SCI doorbell is required upon completion |
| Input Buffer | Byte 0x00 | Control Enable<br>0x00 = This processor's frequency status request is ignored<br>0x01 = This processor's frequency status needs to be evaluated |
| | Bytes 0x01 – 0x03 | Reserved (must be zero) |

| **Output** | | **Value** |
|---|---|---|
| Status Field | | Refer to Section 2.1.6 |
| Output Buffer | Byte 0x00 | Average Frequency Ratio<br>The average , unhalted, frequency of this logical processor, expressed as a percentage of the Nominal CPU Frequency [0-255%] |
| | Byte 0x01 | Current Frequency Limit<br>The maximum frequency the processor is currently allowed to achieve, as a percentage of the nominal CPU frequency [0-254%.] A value of 255 is used to indicate that the CPU is not frequency limited. |
| | Bytes 0x02 – 0x03 | Reserved (must be zero) |

## 3.2 Set Desired Frequency

The Set Desired Frequency command is used to communicate to the platform the desired frequency that the Operating System wants each logical processor's clock to run at. Every logical processor's input and output buffers are evaluated during this command.

| **Input** | | **Value** |
|---|---|---|
| Command Field | Bits 14:0 | X000:0000:0000:0001 |
| | Bit 15 | Set if SCI doorbell is required upon completion |
| Input Buffer | Byte 0x00 | Control Enable<br>0x00 = This processor's frequency control is ignored.<br>0x01 = This processor's frequency control must be evaluated. |
| | Byte 0x01 | Desired Frequency Ratio<br>Desired frequency of the logical processor, expressed as a percentage of the Nominal CPU Frequency [0-255%] |
| | Bytes 0x01 – 0x03 | Reserved (must be zero) |

| **Output** | | **Value** |
|---|---|---|
| Status Field | | Refer to section 2.1.6 |
| Output Buffer | Bytes 0x00 – 0x03 | Reserved (must be zero) |

# 4. Alert Mechanism

The PCC interface has two alerting or “doorbell” mechanisms. The first is used by the Operating System to inform the platform firmware that a command has been sent. The second is optional and may be used by the platform firmware to indicate that the command has been executed.

## 4.1 Operating System to Platform Doorbell

The doorbell used by the Operating System to send a command via the shared memory interface is required and also platform specific. The platform abstracts the method for ringing the doorbell using ACPI. A doorbell consists of a hardware register that is accessed via I/O or memory mapped I/O. ACPI is used to abstract the location of the doorbell and how it should be written. The doorbell specifics are obtained by evaluating the PCCH() object. ACPI does not actually contain code to ring the doorbell. Refer to Section 5 for more information.

## 4.2 Platform to Operating System Doorbell

This optional doorbell used by the platform firmware to indicate that the command has been completed. A bit in the Supported Feature Flags field in the Shared Memory Region Header structure defines whether or not this doorbell is available. Also, the Operating System must indicate that the doorbell on completion is requested by setting bit 15 in the command field. The Operating System’s SCI handler must check the Status Field to determine if the PCC doorbell was the source of the generic SCI interrupt.

# 5. PCC Discovery via ACPI

The PCC Interfaces makes use of ACPI for discovery and abstraction purposes. The interface uses a 4-letter implementation as this is not an official ACPI standard. Platform firmware evaluates _OSC to discover OS support for the PCC interface. If the OS has PCC support enabled then PCC-mode will be **enabled**. Otherwise, if either the platform or the OS fail to support or enable PCC, then PCC-mode is **disabled**. When PCC mode is enabled, it is expected that the platform will not expose processor performance or throttle states via the standard ACPI 2.0/3.0 interfaces (e.g. _PSS, _TSS, and related objects). If these objects exist, the OS must ignore them if PCC-mode is enabled.

- The _OSC interface is used by platform firmware to discover Operating System support for the PCC interface.
- The PCCH() method is used to discover the location of the PCC shared memory region. PCCH() also contains the method for accessing the platform doorbell
- The PCCP() method is implemented for every logical processor and is used to discover the offsets of each processor's input and output buffers.
- The optional, ACPI-SIG defined, _PSD() Method should be used by the Operating System to determine clocking domain dependencies between logical processors.

## 5.1 _OSC Discovery/Enablement

The PCC interface is enabled through evaluation of _OSC in the "_SB" scope. The _OSC interface for PCC is identified by the UUID **639B2C9F-7091-491f-BB4F-A5982FA1B546**. Revision 1 is identified by this specification, consisting of 2 DWORDs including the first DWORD specified by the generic ACPI _OSC definition.

The second DWORD contains the following capabilities bit definition:

| Bit | Description | Value |
|---|---|---|
| 0 | PCC supported | 0 = OSPM does not support controlling processor performance using the PCC interface.<br>1 = OSPM supports controlling processor performance using the PCC interface. |
| 31:1 | Reserved | Must be zero. |

OSPM evaluates _OSC with the PCC supported bit set to indicate it supports the PCC interface. Platform firmware's use of _OSC is optional, and OSPM will attempt to evaluate the remaining ACPI interface methods if _OSC is not present or evaluation fails.

When PCC mode is enabled, the OS will evaluate two control methods to obtain the capabilities and interface details. PCCH() is used to discover the location of the shared memory region and doorbell interface. PCCP() is implemented for each logical processor and is use to obtain the offsets specific to that processors input and output buffers. The

OS will also make use of the standard _PSD() method to obtain clock domain dependencies between logical processors.

# 5.2 PCCH() Object

The PCCH() Object is attributed to the System Board Name Space. Only one PCCH() object exists for a platform. It is defined as follows:

```
Name (PCCH, Package()
{       // Field Name                                   Field Type
        ResourceTemplate() {SharedMemoryRegion},        //QWordMemory
        ResourceTemplate() {PlatformDoorbellAddress},   // Register
        PlatformDoorbellPreserveMask                    // QWordConst
        PlatformDoorbellWriteMask                       // QWordConst
})
```

Shared Memory Region

This is QWORD Address Space describing the physical address range of the reserved memory region used for the shared memory interface.

Platform Doorbell Address

This is a Generic Register Descriptor describing the location of the doorbell.. A write to this address, using the Preserve and Write Masks defined below, rings the doorbell. Reads from the doorbell address have no effect on the doorbell.

Platform Doorbell Preserve Mask

This 64-bit field is a bitmask that should be used when read/modify/writing to the doorbell. The bits that are set in this mask should not be modified when writing.

Platform Doorbell Write Mask

This 64-bit field is a bitmask that should be used when read/modify/writing to the doorbell. The bits that are set in this mask should be set when writing to the doorbell.

# 5.3 PCCP() Object

The PCCP() Object is attributed to the Processor Name Space. One PCCP() object exists for each processor in the platform. It is defined as follows:

```
Name (PCCP, Package()
```

**{ // Field Name Field Type**

**InputBufferOffset, // DWordConst**
**OutputBufferOffset // DWordConst**

})

Input Buffer Offset

This is a 32-bit offset in the shared memory region to this processor's input buffer.

Output Buffer Offset

This is a 32-bit offset in the shared memory region to this processor's output buffer.

# 5.4 _PSD() Object

Processor clocking domain relationships specified by the ACPI standard _PSD method will be honored by OSPM. All processors in a domain will set the same frequency with each Set Desired Frequency command, regardless of coordination type specified in _PSD.

# 6. Terms and Definitions

**Clock Throttling** - An reduction in average clock frequency that is achieved by periodically stopping or gating the clock. The frequency of the clock while it is not stopped or gated remains the same.

**Clock Stretching** - An reduction in average clock frequency that is achieved by changing the frequency of the clock itself. The clock does not stop running, except for non periodic periods of time that are required by the clock generating circuitry (e.g. PLL relock.)

**Doorbell** - An alert, interrupt, exception, or event that is used in a messaging interface to signal that data is ready to be consumed by the endpoint.

**Platform Firmware** - Embedded software that ships with a platform that traditionally is used for platform initialization, hardware abstraction, and system management. Examples include system BIOS, Base Management Controller Firmware, etc.

**Operating System –** for the context of this specification, the term Operating System refers to at least one of the traditional embodiments and includes hypervisors. Examples include, but are not limited to, Windows, Linux, UNIX, and VMware Operating Systems.

**ACPI** - Advanced Configuration and Power Interface. Refer to the ACPI working group website for the latest on the ACPI specification http://www.acpi.info/