White Paper

Performance Monitoring Unit Sharing Guide

Peggy Irelan and Shihjong Kuo

Table of Contents

Acknowledgements

This material has been reviewed and updated in coordination with ecosystem OEMs, OSVs, VMVs, ISVs and Tool providers.

Thanks to all the technical contributors and facilitators for supporting the breath of the reviews.

A special thanks to VMware* for raising attention to this issue and the impact to the ecosystem, and for providing the motivation to start the development of these guidelines.

* Other names and brands may be claimed as the property of others

Terms

Term	Definition
PerfMon	Short for Performance Monitoring. The term used for the current collective monitoring hardware resources e.g. counter/controls etc supported on Intel platforms.
PMU	The Performance Monitoring Unit of processors supporting Intel® 64 and IA-32 architectures. Generally, it consists of collections of MSRs. The collection of MSRs include counter registers, event programming MSRs, global event control MSRs. PMUs of older processors are model-specific; PMU interfaces in more recent processors are evolving towards higher degrees of architectural stability.
MSR	Model Specific Register. PMU counter and counter control registers are implemented as MSR registers. They are accessed via the **RDMSR** and **WRMSR** instruction. Certain counter registers can be accessed via the **RDPMC** instruction. As defined in Volume 2B of the Programmer's Reference Manual. RDPMC is available to software at any privilege level; RDMSR and WRMSR are available only to software running at ring 0.
PEBS	Precise Event Based Sampling. A special counting mode in which counters can be configured to overflow, interrupt the processor, and capture machine state at that point.
CPL and Ring Level	Intel processors operate in privilege levels zero through three. Typically operating system code executes in privilege level 0. User (or privilege levels 1, 2, or 3) refers to less privileged states of execution. User code typically executes at privilege level 3.
In-use	Defines a software convention between software agent's interactions with PMU hardware with respect to the subscription of PMU hardware. A set of PMU hardware (a counter control register, associated counter MSR, or feature) is "in-use" if conditions described in the guidelines section of this document are met.
Agent	Short for PMU agent. A PMU agent is a privileged software process that reads and writes to PMU hardware. Specifically, it could be any one of the following: Firmware (BIOS), OS, hosted VMM, Performance Monitoring tools, ISR, or any other ring 0 software.
First-use Agent	A PMU agent that discovers a set of PMU hardware was not in-use and programs that piece of PMU hardware to in-use state.
ISR	Interrupt Service Routine.
PMI	Performance Monitoring Interrupt. This interrupt is generated when a counter overflows and has been programmed to generate an interrupt, or when the PEBS interrupt threshold has been reached. The interrupt vector for this interrupt is controlled through the Local Vector Table in the Local APIC.
VMX	Virtual Machine eXtensions. Hardware extensions provided to enable x86 hardware virtualization.
SMM	System Management Mode.

About this document

Until fairly recently, the performance monitoring unit (PMU) of Intel® processors employed model-specific programming interfaces and required unique treatment by performance-monitoring users or software agents.

In order to facilitate simpler development and maintenance of performance monitoring tools, Intel has begun the transition to create an architecturally-defined approach for software agents to interacting with the PMU. This approach is known as "architectural Performance Monitoring (PerfMon)," as opposed to the traditional "model-specific Performance Monitoring (PerfMon)." The purpose of architectural PerfMon is to provide a functionally and logically consistent set of capabilities with a consistent hardware interface that developers can rely on now and in the future.

Performance Monitoring (PerfMon) is a limited system resource. While programming control of PMU hardware is available on a first come first serve basis, there is currently no hardware mechanism available to 'reserve' a piece of PMU resource. A software agent can intentionally or unintentionally overwrite previously programmed PMU settings. This situation can be improved with software updates and PMU sharing guidelines between users or agents.

This paper provides a set of guidelines between multiple software agents sharing the PMU hardware on Intel processors. These conventions are not attempting to address PerfMon virtualization but how independent software components (e.g., BIOS and OS; e.g., hosted Virtual Machine Monitor (VMM) and hosting OS) can coordinate use of PerfMon. In the case of PerfMon virtualization (HV and guest), the software is not fully independent, and a HyperVisor/VMM can use different approaches, as long as it uses these conventions globally relative to other independent software (e.g., BIOS). All PerfMon resource definitions are documented in Volume 3B of the *Intel® 64 and IA-32 Architectures Software Developer's Manual.*

Usage and access, beyond Ring 0 programming, is not restricted. These guidelines are for software agents including Firmware (BIOS), VMM, OS and Tools that program and utilize or plan to utilize PerfMon resources.

These guidelines are not setting requirement for PerfMon virtualization under VMX.

All conventions are recommendations. There is no new hardware or hardware enforcement being defined by this paper.

Programmer's Reference Manual Volume 3B and these Guidelines

PerfMon support and usage expectations outlined Programmer's Reference Manual Volume 3B must still be followed:

- **Facility enumeration**: The version identifier is retrieved by querying CPUID.0AH:EAX[bits 7:0]. If the version identifier is greater than zero, architectural performance monitoring capability is supported. Software queries the CPUID.0AH for the version identifier first; it then analyzes the value returned in CPUID.0AH.EAX, CPUID.0AH.EBX to determine the facilities available.
    - Bits 8 through 15 of CPUID.0AH.EAX indicate the number of performance monitoring counters available on the logical processor (each IA32_PERFEVTSELx MSR is paired to the corresponding IA32_PMCx MSR).
    - Bits 0 through 5 of CPUID.0AH.EDX indicate the number of fixed-function performance counters available per thread.
- **Events only supported by enumeration**: A processor that supports architectural performance monitoring may not support all the predefined architectural performance events (Table 18-6). CPUID.0AH:EAX[31:24] indicates events not available.
- **No software precision programming decisions or functionality**: Programming decisions or software decisions for functionality should not be based on the event values or dependent on the existence of performance monitoring events.
- **Known starting state:** Software requires a known starting state. After CPU reset, all counters and control registers are disabled and clear/reset to '0. The only exception to this is the IA32_PERF_GLOBAL_CTRL control MSR, all programmable counter global enable bits are reset to '1.

General PerfMon programming guidelines:

- Disable PerfMon counter(s), using IA32_PERF_GLOBAL_CTRL or IA32_PERFEVTSELx.ENABLE, before programming (writing) the counter (IA32_PMCx) MSRs.
- Utilize kernel preemption disabling facilities provided before reading the in-use indication and programming the counter(s) to reduce race conditions.
- To facilitate sharing agents must move to read-modify-write to access MSRs.

- **An 'in-use' resource is indicated by the following:**
  - **Programmable counter is in-use if the 'Event select field' in IA32_PERFEVNTSELx is non-zero.**
  - **A fixed counter is in-use if its 'Enable field' in IA32_FIXED_CTR_CTRL is non-zero.**
  - **PMI or APIC Performance Monitoring Interrupt Vector is in-use if:**
    - **The 'INT/PMI' bit, bit 20, in any IA32_PERFEVNTSELx is non-zero.**

      or
    - **The 'INT/PMI' bit, the 4th bit in the corresponding fixed counter control field in IA32_FIXED_CTR_CTRL, is non-zero. E.g. If the platform supports 3 fixed counters. The PMI in-use check is IA32_FIXED_CTR_CTRL bits 3, 7 or 11 are non-zero the PMI vector is in-use.**
  - **Model specific feature or event MSRs are in-use if the 'programming/enable' MSR is non-zero.**
    - **For example on** Intel® Core™ i7 processor **:**
      - **PEBS is 'in-use' if (0x3F1) MS_PEBS_ENABLE[3:0] != 0.**
      - **Load Latency feature is 'in-use' if (0x3F1) MS_PEBS_ENABLE[35:32] != 0.**
      - **Extended filter MSRs for the following:**
        - **Offcore Traffic events are 'in-use' if (0x1A6/0x1A7) MS_OFFCORE_REQ0/1 != 0.**
        - **LBR filter is enabled/'in-use' if (0x1C8) MS_LBR_FILTER_SELECT != 0.**
    - ***Note: Model specific PerfMon events/features and the corresponding MSRs are subject to change.***

Sharing Guidelines

Agents must provide a software calling and launch configuration facility to relinquish or release the use of PerfMon resources.

- Agents must be able to be configured and function without PerfMon resources.
- Discontinue or relinquish use of the counter by zeroing the counter control register (IA32_PERFEVNTSELx) or control field and the corresponding counter.
- Agents must discontinue or relinquish PerfMon resource on exit or unload.
- *Note: Tools agents that follow these conventions and utilize counter resources are at risk of being 'killed' by the user such that the tool is not able to cleanly release the counter resources. Tools should maintain a resource usage history and clean exit information file so that if they are launched again they can reclaim the previously used resources.*
    - *If the Tool is not able to cleanly exit, and does not implement a kill-reclaim capability, these resources will remain in-use until reboot. Tools should document the impact to the platform when the user kills the application and does not reboot the system.*

Agents should program the counter at the first opportunity.

- An agent that discovers a counter is in-use, at first opportunity or later, will assume the counter has been claimed by another agent and will not change its programming or disable it.
    - This includes the VMM. The VMM should ensure any counter in-use by firmware is not disabled.
- If an agent has programmed counters at its first opportunity, that agent reserves the right to "reclaim" these counters if the counter, at any point, is reprogrammed by another agent.

Agents should consume a minimal set of resources.

- If at all possible, use a fixed function counter.
- If a programmable counter is required, use the least capable counter.
    - For example, on Intel® Core™ processors and Intel® Core™ Duo processors Precise Event Based Sample (PEBS) is only available on counter 0. If at all possible, agents should avoid using counter 0 to allow PEBS to be utilized by another agent.

If at all possible, configure the fixed counter(s) to be an always running (monotonically increasing) counter(s) and not reprogram.

- Program the counters to count all ring levels and not generate a PMI (performance monitoring interrupt).
- This enables agents to check the corresponding 4-bit counter control for 'free running' configurations for possible read-only sharing.

- For example, Fixed Counter 0 is free running if IA32_FIXED_CTR_CTRL[3:0] = 3, Fixed Counter 1 is free running if IA32_FIXED_CTR_CTRL[7:4] = 3, Fixed Counter 2 is free running if IA32_FIXED_CTR_CTRL[11:8] = 3.

Using Performance Monitoring Interrupts (PMI): If the agents require 'sampling' or enabling PMI, the agent must:

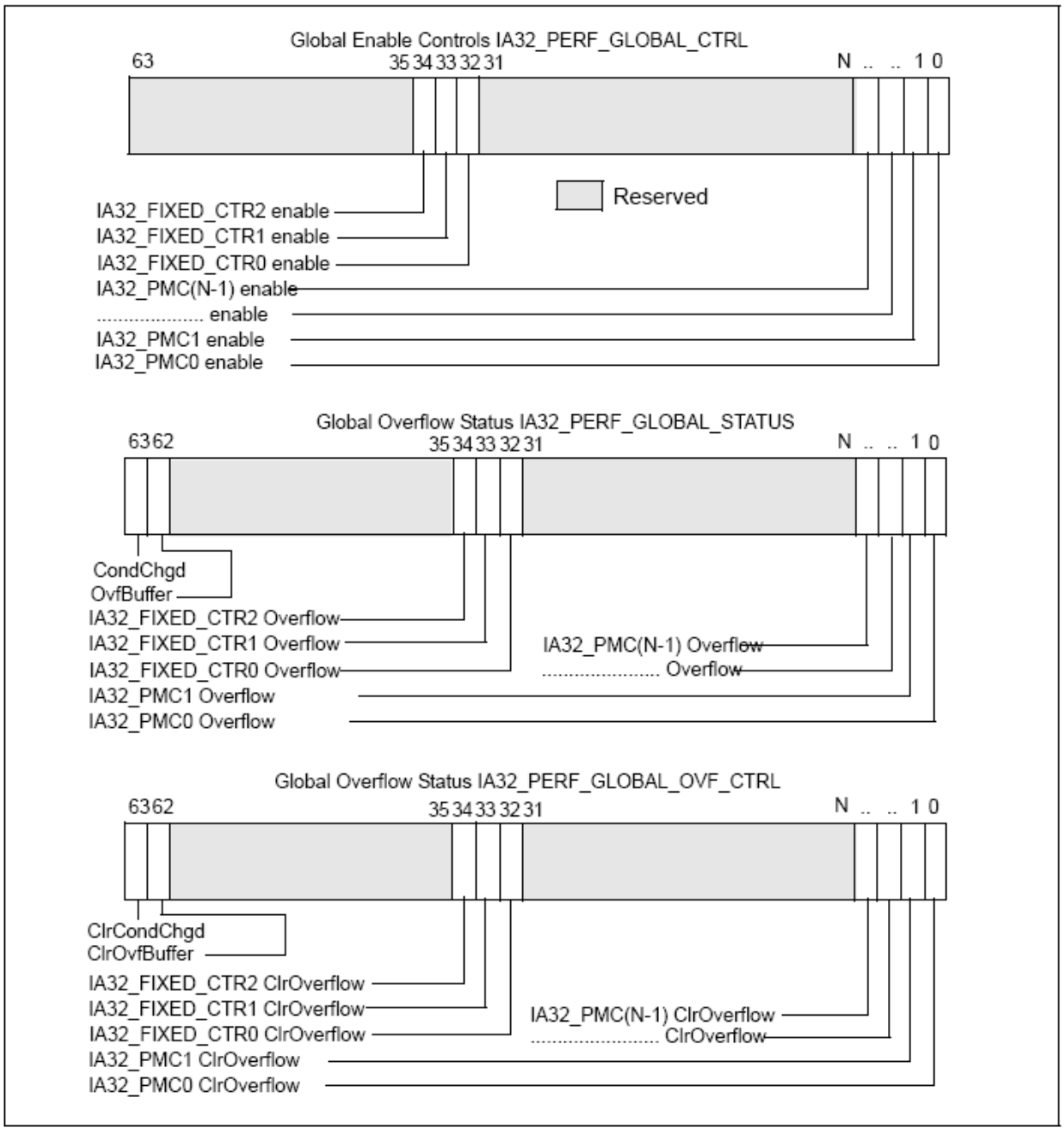
- Poll for an available resource including PMI vector and check for other user(s).
- If other agents are using counters, do not use the 'Freeze PerfMon on PMI' feature in IA32_DEBUGCTL MSR. Setting this bit will disable all counters upon interrupt, possibly negatively impacting other users.
    - Software must utilize the software freeze, read-modify-write to IA32_PERF_GLOBAL_CTRL MSR, in the interrupt service routine and only disable counters under their programming control.

Agents are free to utilize the SMM Freeze feature in IA32_DEBUGCTL MSR. Firmware must not disable this feature while in SMM.

Appendix – MSR Definition(s)

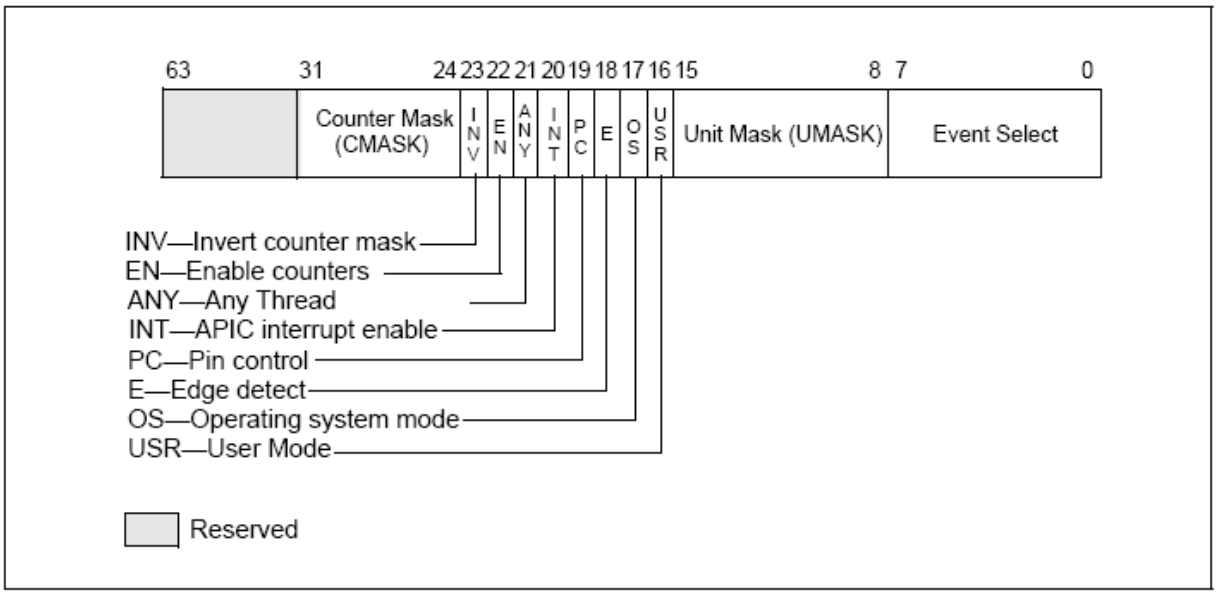
MSR definitions below for Intel® Core™ i7 processor.

Global Control and Status MSR



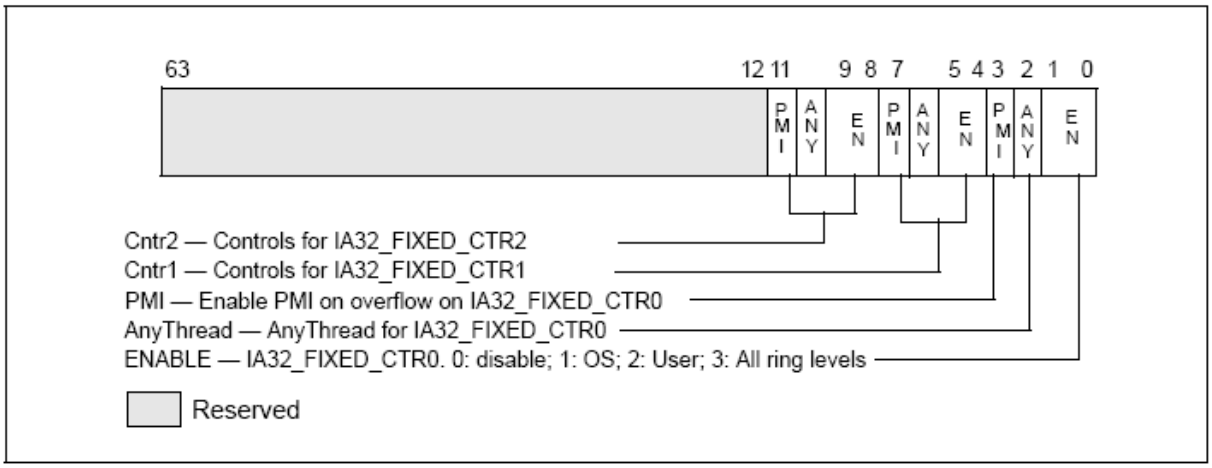
Layout of Global Performance Monitoring Control MSR

IA32_PERFEVNTSELx MSR



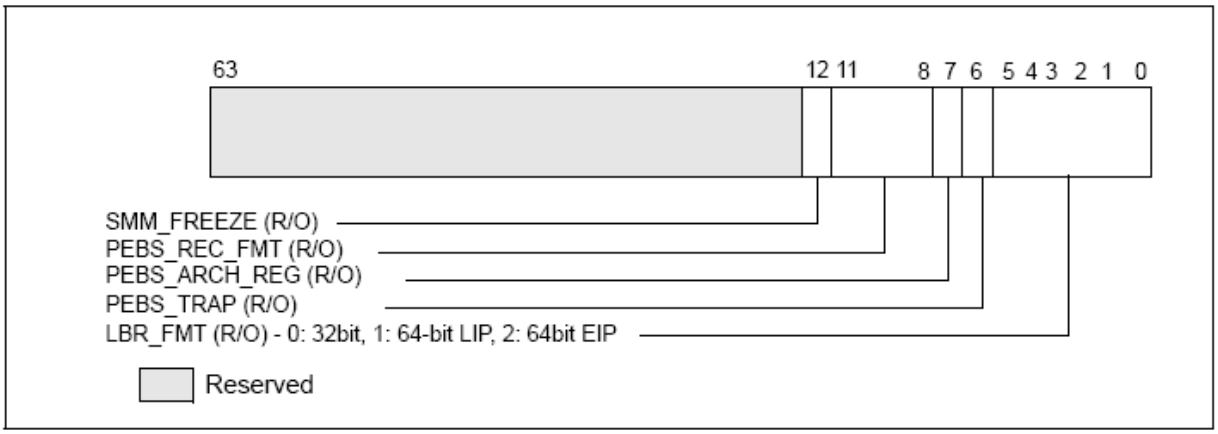
Layout of IA32_PERFEVTSELx MSRs Supporting Architectural Performance Monitoring Version 3

IA32_FIXED_CTR_CTRL MSR



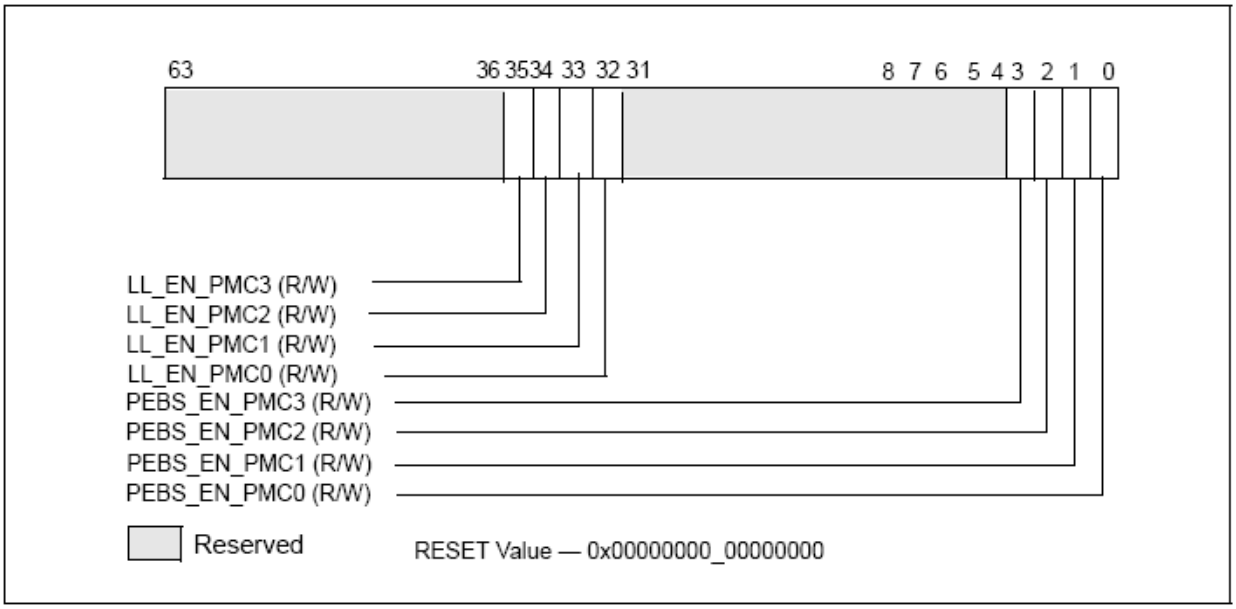
Layout of IA32_FIXED_CTR_CTRL MSR Supporting Architectural Performance Monitoring Version 3

IA32_PERF_CAPABILITIES MSR



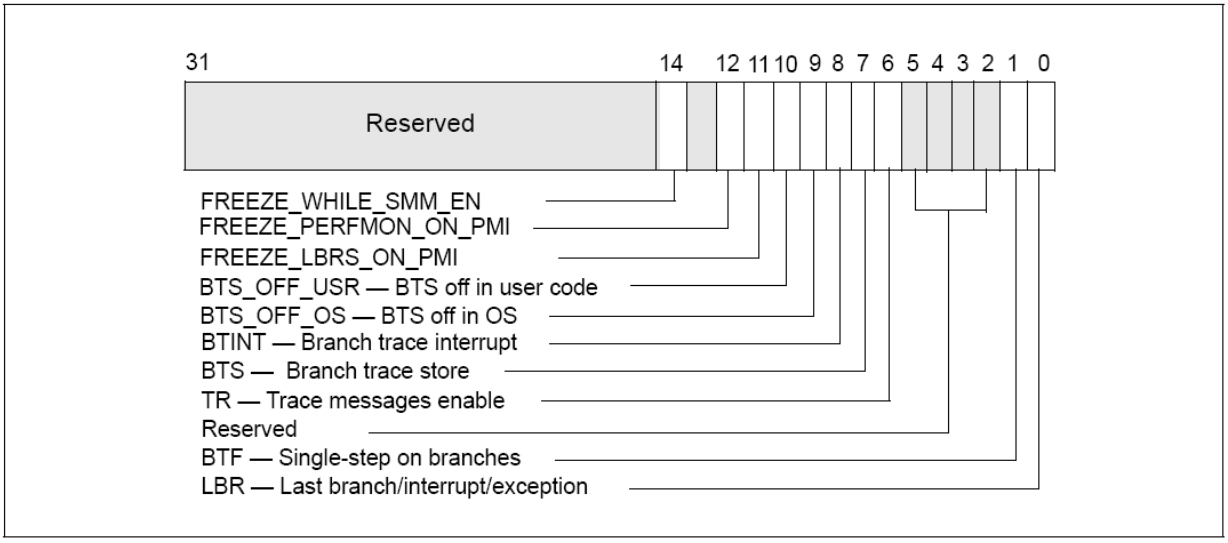
Layout of IA32_PERF_CAPABILITIES MSR

MS_PEBS_ENABLE MSR (Intel® Core™ i7 processor



Layout of PEBS_ENABLE MSR

IA32_DEBUGCTL MSR



IA32_DEBUGCTL MSR for Processors based on Intel Core microarchitecture

About the Authors

Peggy Irelan is a Principal Engineer within the Software & Services Group at Intel. She works on processor observation/monitoring hardware architecture and usage models. She has a particular interest in advancing 'smart software' enabled through hardware feedback, monitoring/sensing. In her spare time, Peggy likes to try new adventures, the latest of which is scuba diving this year reaching PADI Rescue Diver certification while exploring Beqa, Fiji and Moorea, Tahiti. She can be reached at peggy.j.irelan@intel.com

Shihjong Kuo is a technical lead in Server Platform Group. He works on CPU feature documentation and instruction set enabling. He can be reached at shihjong.kuo@intel.com.