

Topology Aware Register and PM Capsule Interface (TPMI)

Introduction

TPMI (Topology Aware Register and PM Capsule Interface) is a flexible, extendable and software-driver-enumerable MMIO interface for Power Management (PM) features. TPMI, planned for future Intel® Xeon® processor generations, is designed using an architectural PCI-e standards-based model so that PM feature support can be provided cleanly as a driver and not as part of the base OS.

Traditional register interfaces for PM features have required changes to CPU hardware and firmware for each generation and are not enumeration friendly for software; requiring Linux kernel driver changes to be upstreamed for each CPU release. Legacy PM interfaces that use Model Specific Registers (MSR) require a userspace process to trap into the kernel to access the MSR rather than allow the MSR to be accessed directly. TPMI utilizes a one-time trap into the kernel (via MMIO registers) when the userspace process mmap's the MMIO space into its address space. After the mmap, the userspace process is enabled through the same instructions as regular memory reads and writes; without traps and associated redirects for each access. This makes TPMI faster than the legacy MSR-based method.

TPMI is designed to PCIe standard which enables: 1) discovery and access of PM registers in hierarchical SOC, and 2) PM features capsulated as a PCI device. This allows for specific PM items such as RAPL interfaces to only be available from the package root punit; while other features and capabilities could have all punits exposed to software. TPMI supports this flexibility without requiring changes to hardware or freezing the definition in hardware.

The remainder of this document provides specification details for TPMI.

VSEC Structure

TPMI is implemented through PCIe VSEC which stands for Vendor-Specific Extended Capabilities. It is a PCI Express spec defined optional functionality where vendor specific capabilities can be exposed from a PCI Express function. Using VSEC allows TPMI to expose multiple capabilities within one PCI function. Consuming a PCI function for each feature/capability that is desired to expose to the OS would be too expensive for the CPU uncore and mesh resources.



VSEC Capability Structure

VSEC Structure:

| 31 | 24 | 16 | 8 | 0 | Bit | Byte |
|------------------------|---------------------------|---|---------------------------|---|------|-------|
| Next Capability Offset | | Capability Version | PCI Express Capability ID | | 0 | 0 |
| VSEC_LEN | | VSEC_REV | VSEC_ID | | 32 | 4 |
| EntrySize | NumEntries | | RESERVED | | 64 | 8 |
| Address | | | | | tBIR | 96 12 |
| Bits | Field | Description | | | | |
| 15:0 | PCI Express Capability ID | This field is a PCI-SIG defined ID number that indicates the nature and format of the Extended Capability. Extended Capability ID for the Vendor-Specific Extended Capability is 000Bh. | | | | |
| 19:16 | Capability Version | This field is a PCI-SIG defined version number that indicates the version of the Capability structure present. Must be 1h for this version of the specification. | | | | |

| | | |
|--------|-------------------------------|---|
| 31:20 | Next Capability Offset | This field contains the offset to the next PCI Express Capability structure or 000h if no other items exist in the linked list of Capabilities. For Extended Capabilities implemented in Configuration Space, this offset is relative to the beginning of PCI-compatible Configuration Space and thus must always be either 000h (for terminating list of Capabilities) or greater than 0FFh. |
| 47:32 | VSEC_ID | This field is a vendor-defined ID number that indicates the nature and format of the VSEC structure. Software must qualify the Vendor ID before interpreting this field. TPMI PM_Features = 0x42. |
| 51:48 | VSEC_REV | VSEC version, set to 0x1. |
| 63:52 | VSEC_LEN | Number of bytes including this field and the CPU capability field, set to 0x10. |
| 79:64 | RESERVED | Reserved |
| 87:80 | NumEntries | Number of entries. Describes the number of feature interface instances that exist in this VSEC space. This is the number of PM features if VSEC_ID is 0x42. |
| 95:88 | EntrySize | Describe the entry size for each interface instance in 32-bit words. This is the size of each PFS entry, set to 0x2. |
| 98:96 | tBIR | Bar indicator register. Indicates which one of the function's base address register is used for the Table Offset |
| 127:99 | Address | Table Offset (TO): Used as an offset from the address contained by one of the function's Base Address register to point to the base of the discovery entry. The lower three tBIR bits are masked off (cleared to 3'b000) by system software to form a 32-bit Qword-aligned offset |

OOBMSM hosts a PCIe function for the TPMI features. Within the function, a single BAR is used to create multiple VSEC structures. Each VSEC can support multiple Punit

interfaces and PM features. The number and size of the interface is captured in the NumEntries and EntrySize part of the PCI header as shown above. All TPMI features share a single VSEC_ID.

The PFS (PM Feature Structure) is a data structure that describes TPMI features. The starting address of PFS is derived from the tBIR and Address fields above. PFS describes all supported TPMI features.

PM Feature Structure

PFS (PM Feature Structure):

| 31 | 24 | 16 | 8 | 0 | Bit | Byte |
|-----------|------------|--|-----------|----|-----|------|
| EntrySize | | NumEntries | TPMI_ID | 0 | 0 | |
| RSVD | | A | CapOffset | 32 | 4 | |
| Bits | Field | Description | | | | |
| 7:0 | TPMI_ID | This field indicates the nature and format of the TPMI feature structure. See the TMPI_ID Encoding table in the following section for additional details. | | | | |
| 15:8 | NumEntries | Number of entries. Describes the number of feature interface instances that exist in the PFS. This represents the maximum number of Punits (i.e. superset chop) of all SKUs. | | | | |
| 31:16 | EntrySize | Describe the entry size for each interface instance in 32-bit words. | | | | |
| 47:32 | CapOffset | Specify the upper 16 bits of the 26 bits Cap Offset (i.e. Cap Offset is in KB unit) from the PM_Features base address to point to the base of the PM VSEC register bank. | | | | |

| | | |
|-------|------------------|---|
| 49:48 | Attribute | Specify the attribute of this feature. 0x0=BIOS. 0x1=OS. 0x2-0x3=Reserved. OS/driver can choose to hide the MMIO region if Attribute=0x0. |
| 63:50 | RSVD | Reserved |

TPMI_ID Encoding

| PM Feature | Encoding | Description |
|------------|----------|---|
| RAPL | 0x0 | Socket RAPL, DRAM RAPL, Platform RAPL, etc. |
| PEM | 0x1 | Power and Performance Excursion Monitors |
| UFS | 0x2 | Uncore Frequency Scaling |
| PMAX | 0x3 | PMax |
| RSVD | 0x4 | Reserved |
| SST | 0x5 | Intel Speed Select Technology |
| MISC_CTRL | 0x6 | Misc control and status registers - package root instance |
| RPLM | 0x7 | Runtime PLL Lock Status Monitor |
| RSVD | 0x8 | Reserved |
| RSVD | 0x9 | Reserved |

| PM Feature | Encoding | Description |
|--------------|----------|------------------------------------|
| FHM | 0xA | FIVR Health Monitor |
| RSVD | 0xB | Reserved |
| PLR | 0xC | Perf Limit Reason |
| BMC_CTL | 0xD | BMC to Primecode mailbox interface |
| RSVD | 0xE-0xFC | Reserved |
| TPMI_CONTROL | 0x80 | TPMI Control Interface |
| TPMI_INFO | 0x81 | TPMI Info Registers |
| CSR_ALL | 0xFD | CSR on all punits |
| CSR_COMPUTE | 0xFE | CSR on compute punit(s) |
| CSR_PKG_ROOT | 0xFF | CSR on package root punit |

TPMI Control Interface

High level guiding principles for software interface:

- If a feature is enabled, OOB is always enabled
- OOB can disable inband
- Inband can't disable OOB

i.e. OOB >= Inband

Boot time platform SW uses TPMI control interface to hide/lock and make interface mutex selection for each individual TPMI feature. Runtime SW uses this interface to discovery the interface selection setting of each TPMI feature.

BMC can use this interface to lock out inband SW write access privilege per TPMI feature. Read access is by default allowed for both inband and out-of-band SW for any enabled TPMI feature. BMC can use this interface to lock out inband SW reads for the allowed features as specified in an allow list. All TPMI PM features are enabled by default in the TPMI control Interface.

SW discovers TPMI control interface as a feature the same way as it would do for all other TPMI PM features. The PFS entry with TPMI_ID=0x80 points to a MMIO address space inside the PCIe device that hosts TPMI control interface registers.

Future generations of Intel® Xeon® Processors® will support command opcodes 0x10 (get_state) and 0x11 (set_state) for TPMI PM features.

Inband SW accesses TPMI control interface through MMIO registers. BMC accesses it through PMT watcher API.

Registers

TPMI control interface registers are described below.

| | | | |
|-----------------------|---------------|----------------|---|
| Name | | | TPMI_CONTROL_STATUS |
| Domain | | | TPMI Register |
| # of Registers | | | 1 |
| Address offset | | | 0x0 |
| Description | | | TPMI watcher mailbox Control and Status register. |
| Bits | Access | Default | Description |
| 0:0 | RW1S | 0 | RUN_BUSY Requester sets this bit to trigger CPU to execute command. CPU clears this flag, when Data and Status fields are updated by it at the end of the command flow. |

| | | | |
|-----------------------|------|---|--|
| 3:1 | RO | 0 | RESERVED Reserved. |
| 5:4 | RO | 0 | OWNER Mailbox ownership - read only: 2'b00 - None - Mailbox is free to use 2'b01 - In-band agent 2'b10 - Out-of-band agent 2'b11 - Reserved |
| 6:6 | RW1S | 0 | CPL Complete bit. This flag shall be set by the agent to indicate when it has finished reading from the mailbox (including control/status). |
| 7:7 | RO | 0 | RESERVED Reserved. |
| 15:8 | RO | 0 | STATUS_CODE Status of TPMI mailbox operation. Updated by CPU at the end of read or write operation. 0x40 - Success 0x90 - Failure 0x80 - Timeout 0x81 - Command Not Serviced |
| 31:16 | RW | 0 | PACKET_LENGTH Length of packet in dwords. For TPMI the length is fixed and equal to 2 dwords. |
| 63:32 | RO | 0 | RESERVED Reserved. |
| Name | | | TPMI_COMMAND_DATA |
| Domain | | | TPMI Register |
| # of Registers | | | 1 |
| Address offset | | | 0x8 |
| Description | | | TPMI mailbox Command register. |

| Bits | Access | Default | Description |
|-----------------------|--------|---------|--|
| 7:0 | RW | 0 | COMMAND TPMI control command issued by the requester. |
| 31:8 | RW | 0 | RESERVED Reserved. |
| 63:32 | RW | 0 | DATA TPMI data written or read by the requester. |
| Name | | | TPMI_CAPABILITIES |
| Domain | | | TPMI Register |
| # of Registers | | | 1 |
| Address Offset | | | 0x10 |
| Description | | | TPMI Capability Enable Bitmask. The supported TPMI_IDs are marked with bit set to 0x1. Out-of-band SW (BMC) can read this register through Watcher API prior to BIOS boots to determine supported TPMI feature list without going through MMIO based PFS table. |
| Bits | Access | Default | Description |
| 255:0 | RO | 0 | CAPABILITIES Bitmask (256 bits) of TPMI_IDs. The supported TPMI_IDs are marked with bit set. |

TPMI_GET_STATE

This command allows software to discover the current state and access privilege of a feature indexed by its TPMI_ID.

Input:

TPMI_COMMAND_DATA.COMMAND

| Field | Description | Bits | Width |
|---------|---|------|-------|
| COMMAND | Specify command = TPMI_GET_STATE (0x10) | 7:0 | 8 |

TPMI_COMMAND_DATA.DATA

| Field | Description | Bits | Width |
|---------|---|------|-------|
| TPMI_ID | Specify the feature TPMI ID | 15:8 | 8 |

Output:

TPMI_COMMAND_DATA.DATA

| Field | Description | Bits | Width | Default |
|----------------|---|-------|-------|---------|
| STATE | 1: The feature as specified by TPMI_ID is enabled. 0: The feature is disabled or not supported. | 0:0 | 1 | 1 |
| IB_WRITE_BLOCK | 0: Inband writes allowed for the feature specified by TPMI_ID. 1: Inband writes blocked. | 4:4 | 1 | 0 |
| IB_READ_BLOCK | 0: Inband reads allowed for the feature specified by TPMI_ID. 1: Inband blocked. | 5:5 | 1 | 0 |
| PCS_SELECT | 0: TPMI. 1=PCS. Indicate which interface will be used by OOB SW for the feature specified by TPMI_ID. | 6:6 | 1 | |
| TPMI_ID | Retains the feature ID from the input | 15:8 | 8 | |
| LOCK | 1=Settings of the feature as specified by TPMI_ID is locked. 0=Unlock. | 31:31 | 1 | 0 |

TPMI_SET_STATE

This command allows software to enable/disable/lock a TPMI feature indexed by its TPMI_ID..

Note that only OOB SW has write access to IB_READ_BLOCK, IB_WRITE_BLOCK and PCS_SELECT.

If a feature is already locked (LOCK==1) then this command returns error when SW writes.

Input:

TPMI_COMMAND_DATA.COMMAND

| Field | Description | Bits | Width |
|---------|---|------|-------|
| COMMAND | Specify command = TPMI_SET_STATE (0x11) | 7:0 | 8 |

TPMI_COMMAND_DATA.DATA

| Field | Description | Bits | Width | IB Permission | OOB Permission |
|----------------|---|------|-------|---------------|----------------|
| STATE | Set the desired state. 1=Enable the feature as specified by TPMI_ID. 0=Disable the feature. When cleared(0), access to any register of that TPMI feature is blocked. | 0:0 | 1 | RW | RW |
| IB_WRITE_BLOCK | 0 (Default) : Inband writes allowed for the feature specified by TPMI_ID. 1: Inband writes blocked. This bit is only writable by Out-of-band SW agent. Inband SW writes to this bit is ignored/dropped. | 4:4 | 1 | RO | RW |
| IB_READ_BLOCK | 0: Inband reads allowed for the feature specified by TPMI_ID. 1: Inband reads blocked. This bit is only writable by Out-of-band SW agent. Inband SW writes to this bit is ignored/dropped. This bit can be set to 1 only if IB_Rd_Blk_Allowed is 1 in the LTM table . | 5:5 | 1 | RO | RW |

| Field | Description | Bits | Width | IB Permission | OOB Permission |
|------------|---|-------|-------|---------------|----------------|
| PCS_SELECT | 0: TPMI. 1=PCS. Indicate which interface will be used by OOB SW for the feature specified by TPMI_ID. This bit is only writable by Out-of-band SW agent. Inband SW writes to this bit is ignored/dropped. | 6:6 | 1 | RO | RW |
| TPMI_ID | Specify the feature ID | 15:8 | 8 | RW | RW |
| LOCK | 1=Lock settings of the feature as specified by TPMI_ID. 0=Unlock. W1S Semantics. Once a feature is locked, its states remain the same till next reset. | 31:31 | 1 | RW | RW |

Output:

- None in TPMI_COMMAND_DATA

Note:

- IB = Inband SW, e.g. BIOS, OS/driver
- OOB = Out of band SW, e.g. BMC SW

Roles and Responsibilities

- OOB SW sets PCS_SELECT to 1 and IB_WRITE_BLOCK to 0 if it wants to use PCS service instead of TPMI.
- OOB SW sets PCS_SELECT to 0 and IB_WRITE_BLOCK to 1 if it wants to control a PM feature using TPMI.
- If PCS_SELECT=IB_WRITE_BLOCK=0, inband SW owns TPMI interface, OOB SW should not attempt to control (write) via TPMI or PCS.
- OOB SW sets IB_READ_BLOCK to 1 if it wants to block inband reads for the features in the allowed list.
- SW can change STATE as long as it is not in locked state. This can be used to program and hide a feature.
- BIOS must lock all TPMI features prior to booting OS.
- If BIOS fails to lock any one of the TPMI features (except TPMI_CONTROL), BIOS shall report error and stop booting.

- SW (including OS/driver, BIOS, BMC) should perform atomic Read-Modify-Write operation when using the TPMI control interface. i.e. SW flow should look like the following:
 1. Get the ownership of the TPMI control interface
 2. Perform TPMI_GET_STATE flow to learn the current setting. Keep the ownership of the interface. i.e. don't set CPL bit until step 5 below.
 3. Modify it to the new desired setting
 4. Perform TPMI_SET_STATE flow
 5. Set the CPL bit to relinquish the ownership.

SW is expected to complete the above 5 steps within 2 seconds.

Address Space Layouts

The diagrams below showing example of several TPMI features. PCIe config space hosts the VSEC structure that contains TPMI VSEC_ID, base address pointer and other info. PFS structure describes each PM feature.

The numbers in the sheet below are for illustrative purposes only.

| PCIe Device (GNR: OOBMSM) | | | | | TPMI Control Interface | |
|---------------------------|-----------|--|--|--|------------------------|---------------------|
| BAR0 | 0x100,000 | | | | Offset | Register Name |
| BAR1 | 0x200,000 | | | | 0 | TPMI_CONTROL_STATUS |
| BAR2 | 0x300,000 | | | | 8 | TPMI_COMMAND_DATA |

| VSEC Capability Structure | | | | |
|---------------------------|-------------|------------|------|---------|
| VSEC_ID | Num Entries | Entry Size | tBIR | Address |
| PM_Features (0x42) | 10 | 2 | 1 | 2000 |
| Telemetry | | | | |

| PM Feature Structure (PFS) | | | | | |
|----------------------------|--------------|------------|------------|-----------|--|
| Num Entries | TPMI_ID | Entry Size | Cap Offset | Attribute | |
| 1 | RAPL | 96 | 4K | OS | |
| 5 | PEM | 6 | 8K | OS | |
| 1 | PMAX | 16 | 12K | BIOS | |
| 1 | DRC | 8 | 16K | OS | |
| 5 | SST | 182 | 20K | OS | |
| 1 | UFS | 8 | 24K | OS | |
| 5 | CSR_ALL | 160 | 28K | OS | |
| 3 | CSR_COMPUTE | 40 | 32K | OS | |
| 1 | CSR_PKG_ROOT | 20 | 36K | OS | |
| 3 | MISC_REGS | 16 | 40K | OS | |

| SW View - Examples | |
|----------------------------|---|
| BAR[1] | 0x200,000 |
| BasePtr of PFS | 0x200,000 + 2000 |
| TPMI_ID = RAPL | |
| BasePtr[RAPL] | 0x200,000 + 2000 + 4K |
| reg_addr[instance, offset] | 0x200,000 + 2000 + 4K + instance*96*4 + offset, Where instance = [0..NumEntries-1] |
| TPMI_ID = PMAX | |
| BasePtr[PMAX] | 0x200,000 + 2000 + 12K |
| reg_addr[instance, offset] | 0x200,000 + 2000 + 12K + instance*16*4 + offset |
| TPMI_ID = CSR_ALL | |
| BasePtr[CSR_ALL] | 0x200,000 + 2000 + 28K |
| reg_addr[instance, offset] | 0x200,000 + 2000 + 28K + instance*160*4 + Toffset where Toffset = offset - StartingOffset |

| | StartingOffset | # registers |
|----------------|----------------|-------------|
| CSR_ALL | 0x110 | 160 |
| CSR_COMPUTE | 0x1B0 | 40 |
| CSR_PKG_MASTER | 0x1D8 | 20 |

TPMI Address mapping

Each TPMI feature may claim a 4KB aligned, could potentially spread into multiple 4KB blocks, region in MMIO address space.

Notes:

- The register mapping of each PM feature is published in EDS.
- Access to the hole(s) in the register space is undefined and may contain any value.

Discovery and Access

TPMI register is MMIO based. Host-based inband software access flow involves multiple steps as described below.

- Search for TPMI's VSEC_ID (0x42) in the VSEC Capability Structure of PCIe devices' extended config space
- Found the TPMI PCIe device and its BARs
- Read tBIR and Table_Offset ("Address") from TPMI VSEC Capability Structure
- Calculate BasePtr, the MMIO base address of PFS table
 - $\text{BasePtr} = \text{BAR}[\text{tBIR}] + \text{VSEC.Table_Offset}$
- Search for the specific TPMI feature of interest in PFS
- Calculate BasePtr[feature], the MMIO base address of the specific TPMI feature
 - $\text{BasePtr}[\text{feature}] = \text{BasePtr} + \text{PFS.Cap_Offset}[\text{feature}]$
- For each TPMI feature, read max SW instance count (PFS.Num_Entries), determine if an instance is valid, and calculate valid SW instance count X as needed.

```
For (p = 0; p < PFS.Num_Entries; p++) {
```

```
// step1: check if this SW instance is valid
```

```
data=[BasePtr[feature] + p * PFS.EntrySize*4 + 0]; // Read
```

```
the first register (offset 0) of this instance
```

```

        valid_instance[p] = (data != -1); // All Fs indicates
invalid instance. Stash this info.

```

```

        // step2: skip the invalid SW instance when accessing TPMI
register

```

```

        If (valid_instance[p]) {

```

```

            // do something with the register

```

```

        } else {

```

```

            continue; // do nothing

```

```

        }

```

```

    }

```

```

    X = popcount(valid_instance[]); // X is the valid SW instance

```

```

count

```

- Calculate MMIO address of the specific register
For features with flat register hierarchy, like RAPL and Pmax, the formula is:
 - $\text{REG_mmio_addr}(\text{instance_idx}, \text{Offset}) = \text{BasePtr}[\text{feature}] + \text{instance_idx} * \text{PFS.EntrySize} * 4 + \text{Offset}$
 - $\text{instance_idx} = 0 \dots \text{PFS.Num_Entries} - 1$
 - "Offset" is the offset from 1st register of the TPMI feature. It is equal to "TPMI_Offset" for flat hierarchy registers.

For features with hierarchical register layout, like SST and UFS, the formula above is guaranteed to get to the 1st register of the feature only.

Formula of the remaining registers is documented in their feature specification.

User is expected to look up the respective feature specification to determine the true offset of 1st register of each TPMI feature. For example, for features like RAPL and SST, their 1st register is at offset 0, while features like CSR_* and RPLM, their 1st register might have non-zero offset.

- Optionally software can build a TPMI address map table to provide MMIO address of each TPMI register of all valid SW instances, or build a SW API that returns MMIO address of each register to caller.
- Inband SW (e.g. BIOS, OS/driver) reads all Fs if a TPMI register is invalid. Out-of-band SW (e.g. BMC) will receive error in PECI completion code (e.g. 0x90) for such cases. Example of invalid register: invalid address or offset, non-existent dielet, out-of-range offset, etc.
-

Access Examples

- **BIOS flow example for configuring Power Limit**
 - TPMI address map table is built in early BIOS boot flow once PCIe BAR is assigned
 - In the Power Limit flow, look up X and MMIO address of PACKAGE_RAPL_LIMIT from the TPMI address map table. X should be 1 for Package-Scoped register.
 - Program PACKAGE_RAPL_LIMIT as usual.
- **BIOS flow example for configuring DESIRED_CORES**
 - TPMI address map table is built in early BIOS boot flow once PCIe BAR is assigned
 - Knowing that DESIRED_CORES is a die-scoped register, so BIOS needs to iterate through multiple instances (X).
 - In the BIOS flow, look up X and MMIO address of DESIRED_CORES from the TPMI address map table.
 - Program DESIRED_CORES as usual.

TPMI Info Registers

TPMI_INFO register block (TPMI ID 0x81) is used by software to link TPMI resources (e.g. register space) to CPU thread. Software uses TPMI_BUS_INFO.package_ID, partition and cd_mask along with MSR 0x54 to map a logical CPU thread to its corresponding TPMI MMIO address.

TPMI_INFO_HEADER

This RO register allows software to discover revision info.

TPMI_INFO_HEADER

| TPMI_INFO_HEADER | | | | | |
|-------------------|------|-------|-------------|--|---------|
| Field Name | Bits | Width | Access Type | Description | Default |
| INTERFACE_VERSION | 7:0 | 8 | RO | Version number for this interface. Bit[7:5] denotes Major Version and bit[4:0] for Minor Version. | 2 |
| RESERVED | 63:8 | 56 | RO | Reserved | 0 |

TPMI_BUS_INFO

This register allows software to discover Segment/Bus/Device/Function, APIC package_ID associated with the package, and bitmap of compute die IDs.

TPMI_BUS_INFO

| TPMI_BUS_INFO | | | | | |
|---------------|-------|-------|-------------|------------------------------------|---------|
| Field Name | Bits | Width | Access Type | Description | Default |
| FUN | 2:0 | 3 | RWL | Function number of the TPMI device | 0 |
| DEV | 7:3 | 5 | RWL | Device number of the TPMI device | 0 |
| BUS | 15:8 | 8 | RWL | Bus number of the TPMI device | 0 |
| PACKAGE_ID | 23:16 | 8 | RWL | The X2APIC Package ID | 0 |

| | | | | | |
|-----------|-------|----|-----|--|---|
| SEGMENT | 31:24 | 8 | RWL | Segment number of the TPMI device | 0 |
| PARTITION | 33:32 | 2 | RWL | Partition number | 0 |
| CD_MASK | 49:34 | 16 | RWL | Bitmap of compute die IDs the partition supports | 0 |
| RSVD | 62:50 | 13 | RWL | Reserved | 0 |
| LOCK | 63:63 | 1 | RWL | When set to 1 the register is locked and becomes read-only until next reset. BIOS writes 1 to this bit after initializing the register | 0 |