# Intel® Trusted Execution Technology (Intel® TXT) DMA Protection Ranges

*Architecture Specification*

*Revision 0.72*

*April 2021*

*Document Number: 633933*

## Notices & Disclaimers

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein.

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

All product plans and roadmaps are subject to change without notice.

The products described may contain design defects or errors known as errata that may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration. No computer system can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

# Contents

1. Introduction ..... 1

2. Intel® TXT Protected Range Registers ..... 2

- 2.1 Register Allocation ..... 3
- 2.2 Usage Protocol ..... 4
- 2.3 Interaction with Special Ranges ..... 4

3. DMA TPR Reporting Structure ..... 5

# Tables

Table 2-1: TPRn_BASE Register ..... 2

Table 2-2: TPRn_LIMIT Register ..... 3

Table 2-3: SERIALIZE_REQUEST Register ..... 3

Table 3-1: DMA TPR Range ..... 5

Table 3-2: TPR Instance Structure (TPR_INS_STRUCT) ..... 6

# Revision History

| Revision Number | Description | Revision Date |
|---|---|---|
| 0.7 | • Initial release | November 2020 |
| 0.71 | • Updated Table 2-1 and Table 2-2 | January 2021 |
| 0.72 | • Tech writer review | April 2021 |

§§

# 1. Introduction

This document describes a new chipset mechanism designed to protect regions of memory from DMA access by primary bus devices. This mechanism is targeted to be used mainly by Intel© Trusted Execution Technology (Intel© TXT) but can be used as a general-purpose DMA protection mechanism in platforms not using Intel TXT.

This mechanism is implemented as several ranges of physical addresses that are protected from DMA access. All ranges can be anywhere in address space. Since the mechanism is primarily targeted to Intel TXT usages, it is called Intel TXT Protected Ranges, or TPRs.

TPRs are defined to replace IOMMMU PMRs and in addition to DMA Protected Range (DPR), which support is continued. More details of the DPR can be found in *Intel© Trusted Execution Technology (Intel© TXT), Software Development Guide, Measured Launched Environment Developer's Guide.*

Advantage of TPRs is that unlike PMRs, TPRs are IOMMU independent. Also, each TPR range is implemented in much smaller number of instances thus removing the need to program symmetrically multiple ranges in multiple IOMMU units.

Other aspects of the TPR usage are almost identical to the usage of PMRs:

The MLE may reside within one or both TPR regions. If the MLE is not within the DPR region then it must be within TPR regions, else SINIT will not permit the environment to be launched.

§§

# 2. Intel® TXT Protected Range Registers

Each TPR is specified by a pair of the registers TPRn_BASE and TPRn_LIMIT.

TPR usage protocol also contains a synchronization step that must be performed by SW or firmware that changes TPRs. This step is accomplished via series of SERIALIZE_REQUEST registers.

Format of all registers follows.

### Table 2-1: TPRn_BASE Register

| Name | | | TPRn_BASE |
|---|---|---|---|
| Description | | | Specifies the start address of TPRn region. TPR region address and size must be with 1 MB resolution. These bits are compared with the result of the TPRn_LIMIT[63:20] applied to the incoming address, to determine if an access fall within the TPRn defined region. |
| Bits | Access | Default | Description |
| 2:0 | RO | 0 | Reserved |
| 3:3 | RO | 0 | RW<br>RW/RO access. 1 – RO register; 0 - RW register. Set to RW for TPRs. |
| 4:4 | RW | 1 | Enable / Disable<br>0 – TPRn address range enabled; 1 – TPRn address range disabled. |
| 19:5 | RO | 0 | Reserved |
| X:20 | RW | 0 | TPRn_BASE_RW<br>Minimal TPRn_Base resolution is 1MB.<br>Applied to the incoming address, to determine if an access fall within the TPRn defined region.<br>**NOTE:** X is determined by a bus width that can be obtained via CPUID function 0x80000008.<br>Bits 19:0 of this register are decoded by hardware as all 0s |
| 63:X | RO | 0 | Unused.<br>**NOTE:** X is determined by a bus width. |

### Table 2-2: TPRn_LIMIT Register

| Name | | | TPRn_LIMIT |
|---|---|---|---|
| Description | | | This register defines an isolated region of memory that can be enabled to prohibit certain system agents from accessing memory. When an agent sends a request upstream, whether snooped or not, a TPR prevents that transaction from changing the state of memory. |
| Bits | Access | Default | Description |
| 2:0 | RW | 0 | Reserved |
| 3:3 | RW | 0 | RW<br>RW/RO access. 1 – RO register; 0 - RW register. Set to RW for TPRs. |
| 19:4 | RO | 0 | Reserved |
| X:20 | RW | 0 | TPRn_Limit_RW<br>Minimal TPRn_Limit resolution is 1MB.<br>These bits define TPR limit address.<br>**NOTE:** X is determined by a bus width that can be obtained via CPUID function 0x80000008. Bits 19:0 of this register are decoded by hardware as all 1s. |
| 63:X | RO | 0 | Unused<br>**NOTE:** X is determined by a bus width. |

### Table 2-3: SERIALIZE_REQUEST Register

| Name | | | SERIALIZE_REQUEST |
|---|---|---|---|
| Description | | | This register is used to request serialization of non-coherent DMA transactions. The operating system shall issue the register before changing of TPR settings (base / size). |
| Bits | Access | Default | Description |
| 0:0 | RO | 0 | STS<br>Status of serialization request<br>0: Idle<br>1: Serialization in progress |
| 1:1 | RW Pulse | 0 | CTRL<br>Control field to initiate serialization.<br>0: Normal<br>1: Initiate Serialization (self-clear to allow multiple serialization requests). |
| 63:2 | RO | - | Unused<br>Values are not specified and must be ignored. |

## 2.1 Register Allocation

The following details apply to the TPR range register allocation:

- Hardware guarantees that TPRn_BASE register in physical address map will always precede TPRn_LIMIT register and both will constitute solid tuple occupying 16 consecutive bytes. This requirement allows operating system access both TPR registers using address of TPRn_BASE register only.
- Hardware fabrics by design may have several transaction traffic routes. In such cases to protect certain address range from DMA traffic identical programming of several control nodes might be required. These nodes will be referred to as TPR instances.
  - Each of the instances will always have the same TPR count.
  - All instances of the same TPR range must be programmed symmetrically.
- Count and location of serialization registers is dependent on topology of fabrics and is not associated with number and location of TPR control registers in any way.

## 2.2 Usage Protocol

Software setting up new TPR range must follow the following three steps:

1. Program base/limit registers to cover memory range to be protected. If more than one instance of TPR exists in the system, all instances must be programmed identically for the desired protections to be enforced.
2. Serialize DMA transactions in-fly, for which all control bits (CTRL) of all instances of SERIALIZE_REQUEST registers must be set to “1”. Doing so will cause status bits (STS) to transition to “1” to indicate “serialization in progress” state. After that Software (SW) must wait till serialization completion is reported by all status bits (STS) to be deasserted back to “0”.<br>No race condition is possible because Hardware (HW) guarantees that the first SW read of STS bit, after SW sets CTRL bit to “1”, will either be”1” for in-progress status or “0” for serialization complete. To avoid accumulation of waiting times, SW is recommended to execute entire process in two loops: set all CTRL bits in one loop; proceed to waiting of all STS bit de-assertion in another loop.
3. Flush all core and device caches by executing CLFLUSH instruction for entire new protected range.

## 2.3 Interaction with Special Ranges

TPRs must not overlap each other and special security HW ranges existing in the system. Specifically, they must not overlap DMA Protected Range (DPR), IOMMU PMR protected ranges, Isolated Memory Ranges (IMR), and Memory-Mapped I/O ranges. Overlap behavior is model-specific and might be Machine Checks in some cases but in no cases will allow DMA to any address contained in any of the overlapping regions. As a rule – TPRs are intended to protect usable DRAM areas only.

§§

# 3. DMA TPR Reporting Structure

The operating system must know the whereabouts of the TPR Range Registers.

The system firmware (BIOS) is responsible for detecting the TPR hardware in the platform, and for locating the memory mapped TPR registers in the host system address space.

The system firmware must report the base addresses of the TPR register blocks and synchronization registers to system software through the DMA TXT Protected Range (DTPR) ACPI table described below.

System firmware must detect and correctly report number of TPRs, number of TPR instances, and number of synchronization registers based on HW configuration; then construct the DTPR table accordingly. Doing so may require system firmware to construct DTPR table dynamically during boot.

### Table 3-1: DMA TXT Protected Range (DTPR)

| Field | Byte Length | Byte Offset | Description |
|---|---|---|---|
| Header | | | |
| Signature | 4 | 0 | "DTPR". Signature of the DMA TXT Protected Range description table. |
| Length | 4 | 4 | Length, in bytes, of the description table. |
| Revision | 1 | 8 | 1 |
| Checksum | 1 | 9 | Entire table must sum to zero. |
| OEMID | 6 | 10 | OEM ID |
| OEM Table ID | 8 | 16 | For DTPR description table, the Table ID is the manufacturer model ID. |
| OEM Revision | 4 | 24 | OEM Revision of DTPR Table for OEM Table ID. |
| Creator ID | 4 | 28 | Vendor ID of utility that created the table. |
| Creator Revision | 4 | 32 | Revision of utility that created the table. |
| Flags | 4 | 36 | Control flags. Reserved for future |
| Instance Count (InsCnt) | 4 | 44 | Number of HW instances of the same TPR.  All instances of the same TPR must be programmed identically. |
| Instance Array [InsCnt] | InsCnt x sizeof (TPR_INS_STRUCT] | 48 | Array of HW instance structures (TPR_INS_STRUCT) of the same TPR |
| Serialization register Count (SrlCnt) | 4 | - | Number of serialization registers. In specific designs, this count could be zero. |
| TPR Serialization Array [SrlCnt] | SrlCnt x 8 | - | Array of physical addresses of TPR serialization registers. If SrlCnt equals zero, this field is absent. |

### Table 3-2: TPR Instance Structure (TPR_INS_STRUCT)

| Field | Byte Length | Byte Offset | Description |
|---|---|---|---|
| Flags | 4 | 0 | Instance specific control flags. Reserved for future |
| TPR Count (TprCnt) | 4 | 4 | Number of TPR ranges in a single instance. Each instance has the same TPR count. It will always be 2 or more. |
| TPR Array [TprCnt] | TprCnt x 16 | 8 | Array of physical addresses of TPR control register pairs: TPRn_Base/TPRN_Limit in a single TPR instance |

§§