ARM® Architecture Reference Manual Supplement Statistical Profiling Extension, for ARMv8-A

Document number: DDI 0586A

Architecture Reference Manual Supplement Statistical Profiling Extension

Release information

The following changes have been made to this document.

Date	Issue	Confidentiality	Change
29 March 2017	A	Non-Confidential	First Issue

Proprietary Notice

Confidentiality Status

Product Status

The information in this document is final, that is for a developed product.

Web Address

http://www.arm.com

Contents

Preface

This document describes the Statistical Profiling Extension, an optional extension for implementations of the ARMv8-A architecture profile ARMv8.1 or later, and must be read in conjunction with the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile.*

References

This document refers to the following documents and links.

Reference	Document number	Author	Document name
[1]	ARM DDI 0487	ARM	*ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*
[2]	ARM-DEN-0029	ARM	*ARM® Server Base System Architecture*

Feedback

ARM welcomes feedback on its documentation.

If you have comments on the content of this manual, send an e-mail to errata@arm.com. Give:

- The title.
- The document and version number, ARM DDI 0586A.
- The page numbers to which your comments apply.
- A concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

1 Introduction

This section introduces the Statistical Profiling Extension, a mechanism for profiling software and hardware using randomized sampling.

The Statistical Profiling Extension is an optional extension to the ARMv8 architecture, and is introduced as part of ARMv8.2. An implementation that includes the Statistical Profiling Extension must be compliant with at least the ARMv8.1 architecture profile.

The Statistical Profiling Extension introduces a new barrier instruction, the Profiling Synchronization Barrier (`PSB CSYNC`), which is described in detail in the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile.*

The Statistical Profiling Extension introduces a set of registers that are specific to the Statistical Profiling Extension architecture. These registers are accessed as System registers and are described in detail in this supplement.

In addition, the Statistical Profiling Extension adds a limited number of fields to the following ARMv8-A System registers in AArch64 state:

- ID_AA64DFR0_EL1.PMSVer
- MDCR_EL2.{TPMS, E2PB}
- MDCR_EL3.NSPB

For details about these changes, see the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*.

1.1 Statistical profiling

Statistical profiling is a four stage process.

1. An operation is chosen from a sample population, at a programmable interval that might have some random, or pseudorandom, perturbation.
2. A trace of the sampled operation is taken. This includes the PC, events, timings, and data addresses, related to the sampled operation. This is the profiling operation.
3. Before a sample record is created, it is possible to filter out potential sample records generated by the profiling operation by reference to, any or all of the following:
    a. The type of operation.
    b. The Exception level.
    c. Event and latency.
4. A sample record is created that contains the traced information. Sample records that meet the criteria of the filter are written to and stored in a memory buffer. These sample records can be processed by software when the memory buffer is full.

2 ARMv8.2 Statistical Profiling Extension

This section gives an overview of the Statistical Profiling Extension. Sections 3 and 4 describe the programmers' model in detail:

- Programmers' Model Overview.
- Detailed Programmers Model.

2.1 Non-invasive behavior

Statistical Profiling is a non-invasive debug operation:

- While profiling is enabled, the operation and performance of the processing element (PE) must not be significantly impacted between sampled operations, that is, other than for writing out sample records and processing Profiling Buffer management events.
- The performance of the sampled operation and the performance of the PE in general must not be significantly impacted. The sample records are not written to memory until after the sampled operation has completed. However, this does not apply when the user selects a collection of physical addresses for data access operations. In this case, the impact is IMPLEMENTATION DEFINED.
- The profiling operation to write sample records must not be excessively impactful on the performance of the sampled operation or the performance of the PE generally.

2.1.1 Disabled in AArch32

Statistical profiling is disabled in AArch32 state.

2.2 Defining the sample population

All samples are taken from a *population* of *operations*. The population is *dynamic* rather than *static*. That is, if a program executes the same operation multiple times (for example, because of loops and subroutines) then that operation appears multiple times in the population.

The operations are an IMPLEMENTATION DEFINED choice between:

- Architecture instructions.
- IMPLEMENTATION DEFINED microarchitectural operations (*µ-ops*).

Architecture instruction means a single instruction that is defined by the ARMv8 instruction set architecture in AArch64 state.

An architecture instruction might create one or more µ-ops at any point in the execution pipeline. The definition of a µ-op is implementation specific. An architecture instruction might create more than one µ-op for each instruction. A µ-op might also be removed or merged with another µ-op in the execution stream, so an architecture instruction might create no µ-ops for an instruction.

Any arbitrary translation of architecture instructions to an equivalent sequence of µ-ops is permitted. In some implementations, the relationship between architecture instructions and µ-ops might vary over time.

—— Note ——

Sampling from architecture instructions does not require that the instruction is architecturally executed.

2.2.1 Operation sampling

A sample operation is as follows:

1. A sampling interval is written to a sample interval counter by software. The interval is measured in operations.
2. The sample interval counter is decremented by hardware for each operation when sampling is enabled.
3. When the sample interval counter reaches zero, then:
    a) If random perturbation is enabled, the PE continues to count for a random number of further operations while sampling is enabled.
    b) An operation is chosen for profiling.

    Note

    The choice of operation around the sampling point is arbitrary. The chosen operation might be the operation for which the sample interval counter reached 0, or the next operation. The choice must be applied consistently so as not to introduce sampling bias.

4. The sample interval counter is reloaded and the process loops to step 2. It is IMPLEMENTATION DEFINED whether the sample interval counter is reloaded before step 3.a) or at step 3.b). That is, before or after counting the random number of further operations. See Controlling when an operation is sampled .
5. The chosen operation is marked as the sampled operation. The PE collects information about the sampled operation as it executes by a profiling operation.
6. When the sampled operation is completed, the sample record is captured.

The lifespan of a sampled operation completes when any of the following is true:

- The instruction or µ-op is committed to the architectural state of the PE (including completion by generating a synchronous exception).
- The speculative instruction or µ-op is discarded because of misspeculation.
- The instruction or µ-op is replayed, on some microarchitectures.

  For more information on misspeculation and replaying instructions or µ-ops see Operations that might be excluded from the sample population.

2.2.2 Random number generation

The random number generator is IMPLEMENTATION DEFINED. Implementations might use a pseudorandom number. The random number generator must be reset into a useable state. An implementation might include IMPLEMENTATION DEFINED registers to further configure the random number generator.

2.2.3 Defining where operations are sampled

The exact point in the sampled lifetime of operations at which sampled operations are chosen for profiling is IMPLEMENTATION DEFINED.

Note

ARM recommends that the point at which operations are sampled is linked to the definition of the Performance Monitoring Extension (PMU) STALL_FRONTEND and STALL_BACKEND events, so that sampling records information for STALL_BACKEND stalls.

2.2.4 Sample collisions

The maximum number of sampled operations that a PE can support simultaneously in flight is IMPLEMENTATION DEFINED. If the maximum number of sampled operations is in flight at the point when a new sample operation must be taken, the new sample is said to have *collided* with a previous sampled operation.

The PE records the fact that a sampled operation has collided with another sampled operation. Software can also count the number of collisions and gauge the impact of the collisions.

See Sample collision behavior.

2.2.5 Operations that might be excluded from the sample population

It is IMPLEMENTATION DEFINED whether each of the following operations is part of the sample population:

- Operations on misspeculated paths.
- Operations (specifically μ-ops) that do not relate to any architecture instruction.
- Operations that generate non-architectural exceptions.

If the operation is not part of the sample population, the operation does not cause the sample interval counter to decrement, is not counted by the SAMPLE_POP event and therefore is never sampled.

If the operation is part of the sample population, the operation causes the sample interval counter to decrement, is counted by the SAMPLE_POP event, and might be sampled and counted by the SAMPLE_FEED event.

See also Sample Records that might be incomplete.

2.3 Collected data

Unless otherwise stated all sample records that are generated by a profiling operation contain:

- A timestamp, if enabled. This is one of:
  - CNTPCT_EL0.
  - CNTVCT_EL0.

  It is IMPLEMENTATION DEFINED how this timestamp relates to the sampled operation. It might be the time when the sampled operation was taken or any later time during the lifetime of the sampled operation, that is, up to the time when the sampled operation is completed.

  If the Generic Timer system counter is disabled and timestamps are enabled, then it is IMPLEMENTATION DEFINED whether:

  - The Statistical Profiling Extension behaves as if timestamps are disabled.
  - The timestamp that is collected in the sample record is UNKNOWN.

—— **Note** ——

This behavior describes when CNTEN.EN is cleared to 0. This behavior does not apply when the Generic Timer system counter is enabled but not accessible at the current Exception level.

- The context, if enabled, which is one or more of:
  - CONTEXTIDR_EL1.
  - CONTEXTIDR_EL2.
  - The Exception level.
  - The Security state.
- Information about whether the sampled operation generated an exception:
  - The target address for an exception generating operation is not collected.
- Information about whether the sampled operation completed execution.

If the sampled operation completes execution and does not generate an exception, the sample record also contains:

- The PC virtual address for the sampled operation.
- Information about whether the sampled operation is a branch, a load, a store, or other.
- Information about whether the sampled operation is conditional, conditional select, or not.
- The total latency, a cycle count of the lifetime of the sampled operation.
- The issue latency, a cycle count from the start of the lifetime of the sampled operation up to the point when the sampled operation starts executing. A sampled operation might be delayed, for example because the input operands were not available.

The architecture defines a set of additional data that is collected in the sample record for each sampled operation. This is described in the following subsections, and comprises:

- Events.
- Cycle counters. Cycle count values as described in this architecture, which for a particular implementation are fixed with an IMPLEMENTATION DEFINED value, might be omitted from the sample record.
- Addresses.

In addition, the architecture permits IMPLEMENTATION DEFINED events, counters, and addresses to be collected.

2.3.1 Additional information for each profiled branch or exception return

For a completed branch or exception return sampled operation, the profiling operation must record:

- The sampled operation type as an unconditional branch or a conditional branch. Sampled exception returns are treated as unconditional branches by the Statistical Profiling Extension.
- The target virtual address of the branch. The target virtual address includes the Exception level and Security state of the target.

—— **Note** ——

If the sampled operation is an illegal exception return, it is IMPLEMENTATION DEFINED whether the context information recorded in the target virtual address is the actual target context, or the target context that is described by the SPSR.

——————

- If the PE implements branch prediction, whether the branch was correctly predicted or mispredicted.
- Whether the branch was taken or not taken.
- Whether the branch was direct or indirect.

See also Additional information for each profiled conditional instruction (including conditional branches).

—— **Note** ——

A sampled operation that generates an exception is not treated as a branch.

——————

2.3.2 Additional information for each profiled memory access operation

For a completed load, store, or atomic sampled operation that does not generate an exception, the profiling operation must record:

- The data virtual and, if enabled, physical addresses being accessed.
  - If the applicable Top Byte Ignore (TBI) bit is set to one, the virtual address includes any top-byte tag.
  - The physical address is the address the PE accesses in the physical address space, and so includes the Secure Address Space Identifier.
- The sampled operation type, which includes:
  - Whether the sampled operation is a load, store, or atomic.
  - Whether the sampled operation is Load-Exclusive, Store-Exclusive or Load-acquire, Store-release.
  - Whether the sampled operation accesses the general-purpose or SIMD&FP registers.
- The translation latency, a cycle count from the virtual address being generated to the physical address being returned by the MMU.
- Whether the sampled operation accessed the Level 1 data cache and resulted in a hit.
- Whether the sampled operation accessed the data TLB and resulted in a hit.
- An optional, IMPLEMENTATION DEFINED, record of whether the sampled operation accessed Last Level data cache and resulted in a hit.
- An optional, IMPLEMENTATION DEFINED, record of whether the sampled operation accessed *another socket* in a multi-socket system.
- An optional, IMPLEMENTATION DEFINED, indicator of the data source for a load.

For each of the Last level cache and another socket indicators, it is IMPLEMENTATION DEFINED whether this information is present only for load accesses, only for store accesses, for neither, or for both.

—— Note ——

A store might be marked as not accessing a cache or another socket because it completed before doing so. For example, the write was held in a write buffer. This behavior is IMPLEMENTATION DEFINED, and software must be cautious when interpreting such events.

If architecture instructions are sampled, then, for a sampled load/store operation that is not single-copy atomic, the data addresses are the lowest address that is accessed by the sampled operation.

Otherwise the information is for the µ-op that is sampled.

—— Example ——

If an architectural load instruction is split into an address generation µ-op and a load µ-op, then when generating the sample record and filtering based on operation type:

- If the address generation µ-op is sampled, the sampled operation is treated as *other*.
- If the load µ-op is sampled, the sampled operation is treated as a load.

For more information see Filtering sample records and Operation Type packet.

2.3.3 Additional information for each profiled conditional instruction (including conditional branches)

For a completed conditional branch, conditional select, conditional move, or conditional increment sampled operation, the profiling operation must record:

- That the sampled operation was conditional.
- Whether the condition passed or failed.

If a conditional instruction fails its condition code test, it is IMPLEMENTATION DEFINED whether any of the information for the sampled operation that is generated by executing the sampled operation is recorded. Any value that is recorded and is the result of executing the sampled operation is UNKNOWN.

—— Example ——

A conditional branch operation fails its condition code test. It is IMPLEMENTATION DEFINED whether the sample record contains a branch target Address packet. If the sample record contains a branch target Address packet, the value in the packet is UNKNOWN.

2.3.4 Additional information for other operations

For cache maintenance operations by virtual address, cache prefetch, or address translation instructions, the profiling operation:

- Captures an IMPLEMENTATION DEFINED subset of the information captured for a load instruction.
- Treats the operation type as *other* when generating the sample record and filtering based on operation type.

See Filtering sample records and Operation Type packet.

2.3.5 Sample Records that might be incomplete

It is IMPLEMENTATION DEFINED whether the sample record for a misspeculated operation or an operation generating a non-architectural exception, if sampled, is written to the Profiling Buffer:

- If a sample record of a misspeculated operation or an operation generating a non-architectural exception is written to the Profiling Buffer, then neither event #0 (generated exception) nor event #1 (architecturally retired) are set in the record Events packet.
- If a sample record of a misspeculated operation or an operation generating a non-architectural exception is not captured into the Profiling Buffer, then no event packets are output and the sample is not counted by the SAMPLE_FILTRATE event.

Note

It is IMPLEMENTATION DEFINED whether such operations can be sampled. See Operations that might be excluded from the sample population.

All sample records written to the Profiling Buffer contain the Events packet and either the End packet or the Timestamp packet.

If the sampled operation generates an exception, it is UNPREDICTABLE whether the sample record contains any other information.

Where a sampled operation generates an exception and the type of exception means that a particular item is not computed by the sampled operation, that information is not collected by the profiling operation. See Synchronization and Statistical Profiling.

Example

A sampled operation generates a Translation Fault. The physical address for the sampled operation was not generated by the MMU and cannot be recorded.

2.4 Enabling sampling and filtering sample records

Profiling can be disabled at individual Exception levels, and is always disabled at EL3.

When profiling is disabled, no sample records are collected and the sample interval counters retain their values and do not decrement.

Collected sample records can also be filtered by type, by event, and by latency:

- By type, a combination of any or all of:
  - Branches.
  - Loads.
  - Stores.
- By event, a combination of any or all of:
  - Mispredicted branches.
  - TLB missing accesses.
  - Level 1 cache refilling accesses.

  - Architecturally retired instructions.
- By latency:
  - Whether the total latency of the operation exceeds a programmable threshold.

When filtering is enabled, sample records that do not meet the filtering criteria are discarded and are not written to the Profiling Buffer.

2.5 Capturing records of sampled operations

Sample records are written by hardware in a packetized format to a buffer in memory:

- The start and end pointers for this buffer are specified in System registers.
- The pointers are virtual address pointers.
- The PE generates a Profiling Buffer management event when the Profiling Buffer is full.

For more information see The Profiling Buffer.

2.6 PMU Extensions

Two sets of additional events are defined for the PMU. If the Statistical Profiling and Performance Monitoring Extensions are implemented, then the events that are listed in Table 1 must be implemented.

Table 1: Statistical Profiling PMU events

Event number	Event mnemonic	Description
`0x4000`	SAMPLE_POP	**Sample Population** The counter increments for each operation that might be sampled, whether or not the operation was sampled. Operations that are executed at an Exception level or Security state in which profiling is disabled are not counted.
`0x4001`	SAMPLE_FEED	**Sample taken** The counter increments each time the sample interval counter reaches zero and is reloaded, and the sample does not collide with a previous sample. Samples that are removed by filtering, or discarded, and not written to the buffer are counted. The number of operations that might be sampled is counted by SAMPLE_POP, meaning these two events can determine the sampling ratio.

`0x4002`	SAMPLE_FILTRATE	**Sample taken and not removed by filtering** The counter increments each time that a completed sample record is checked against the filters and not removed. Sample records that are not removed by filtering, but are discarded before being written to the Profiling Buffer because of a Profiling Buffer management event, are counted. The number of operations that are sampled is counted by SAMPLE_FEED meaning that: • If all filtering is disabled, then SAMPLE_FILTRATE counts the same as SAMPLE_FEED. • These two events can determine the proportion of sampled operations that met the filtering criteria.
`0x4003`	SAMPLE_COLLISION	**Sample collided with a previous sample** The counter increments for each sample record that is taken when the previous sampled operation has not completed generating its sample record.

Note

These events are discoverable through a read of PMCEID0_EL0[35:32].

The Statistical Profiling Extension also introduces the events that are listed in Table 2. These events can be included in any implementation of the PMU Extension, even if the Statistical Profiling Extension is not implemented.

Table 2: Additional PMU events

Event number	Event mnemonic	Description
`0x0031`	REMOTE_ACCESS	**Access to another socket in a multi-socket system** The counter counts each memory read operation or memory write operation that causes an access to another socket in a multi-socket system. It is IMPLEMENTATION DEFINED whether an access that causes a snoop into another socket but does not return data from or pass data to the remote socket is counted.
`0x0032`	LL_CACHE	**Last Level cache access** The counter counts each memory read operation or memory write operation that causes a cache access to at least the Last Level data or unified cache. Cache maintenance instructions do not count as events.

`0x0033`	LL_CACHE_MISS	**Last Level cache miss** The counter counts each memory-read operation or memory write operation that causes a cache access to at least the Last Level data or unified cache, but is not completed by the Last Level cache. That is, either of the following: • A memory read operation that does not return data from the Last Level cache. • A memory write operation that does not update the Last Level cache. The counter does not count operations that are completed by a cache above the Last Level cache. The number of Last Level cache accesses is counted by LL_CACHE, meaning that these two events can determine the Last Level cache miss ratio.
`0x0034`	DTLB_WALK	**Access to data TLB that causes a translation table walk** The counter counts each memory read or memory write operation that causes a refill of a data or unified TLB involving at least one translation table walk access. This includes each complete or partial translation table walk that causes an access to memory, including to data or translation table walk caches. The number of TLB accesses is counted by the L1D_TLB event, meaning that these two events can determine the data TLB miss ratio.
`0x0035`	ITLB_WALK	**Access to instruction TLB that causes a translation table walk** The counter counts each instruction memory access that causes a refill of an instruction TLB, involving at least one translation table walk access. This includes each complete or partial translation table walk that causes an access to memory, including to data or translation table walk caches. The number of TLB accesses is counted by the L1I_TLB event, meaning that these two events can determine the instruction TLB miss ratio.
`0x0036`	LL_CACHE_RD	**Last level cache access, read** As LL_CACHE, but counts only memory read operations.
`0x0037`	LL_CACHE_MISS_RD	**Last level cache miss, read** As LL_CACHE_MISS_RD, but counts only memory read operations. The number of Last Level cache accesses is counted by LL_CACHE_RD, meaning that these two events can determine the Last Level cache read miss ratio.

`0x0038`	REMOTE_ACCESS_RD	**Access to another socket in a multi-socket system, read** As REMOTE_ACCESS, but counts only memory read operations. It is possible to calculate the number of memory write operations by deducting the number of REMOTE_ACCESS_RD events from REMOTE_ACCESS events.

—— Note ——

These events are discoverable through a read of PMCEID1_EL0[24:17] and PMCEID1[24:17].

2.7 Interaction with the Embedded Cross Trigger

When a sample completes and is written out, CTI input trigger 2 is asserted. This trigger might also be directly connected to other IMPLEMENTATION DEFINED debug features.

2.8 Multithreaded implementations

In a multithreaded implementation, Statistical Profiling is implemented per-thread.

The sample interval counter counts only operations for the thread that is being profiled.

Latency and other cycle counters count each cycle for the PE for which the thread was active and could issue an operation.

The architecture does not define features for inter-thread profiling and does not support sharing the Profiling Buffer between threads.

—— Note ——

An implementation is described as *multi-threaded* when the lowest level of affinity consists of logical processors that are implemented using a multi-threading type approach. That is, the performance of processors at the lowest affinity level is very interdependent. On such an implementation, the value of MPIDR_EL1.MT, when read at the highest implemented Exception level, is 1.

3 Programmers' Model Overview

Profiling is controlled by System registers. There is no direct EL0 access to the control registers and by default profiling is disabled. The programmers' model is divided into the following parts:

- Sampling controls:
  - Controlling when an operation is sampled .
  - Enable and Filtering controls.
  - Controlling the data that is collected.
- Profiling Buffer controls:
  - The Profiling Buffer.
  - Profiling Buffer management .
- Synchronization and Statistical Profiling.

3.1 Controlling when an operation is sampled

The sample interval counter, PMSICR_EL1.COUNT controls when an operation is selected for sampling. In some implementations, a secondary sample interval counter, PMSICR_EL1.ECOUNT, is also used.

The following sections describe the operation of the sample interval counters.

Details of the random or pseudorandom number generator used when PMSIRR_EL1.RND is set to 1 are IMPLEMENTATION DEFINED. See Random number generation.

3.1.1 Initializing one or more sample interval counters

When the PE moves from a state where profiling is disabled to a state where profiling is enabled:

- If PMSICR_EL1 is nonzero, then sampling restarts from the current values in PMSICR_EL1.
- If PMSICR_EL1 is zero, then it is loaded with an initial value. The behavior depends on PMSIRR_EL1.RND and an IMPLEMENTATION DEFINED choice discoverable by a read of PMSIDR_EL1.ERnd.

  If PMSIRR_EL1.RND is cleared to 0:

  - PMSICR_EL1.COUNT[31:8] is set to PMSIRR_EL1.INTERVAL.
  - PMSICR_EL1.COUNT[7:0] is set to `0x00`.

  If PMSIRR_EL1.RND is set to 1 and PMSIDR_EL1.ERnd is 0:

  - PMSICR_EL1.COUNT[31:8] is set to PMSIRR_EL1.INTERVAL.
  - PMSICR_EL1.COUNT[7:0] is set to a random or pseudorandom value in the range `0x00` to `0xFF`.

  If PMSIRR_EL1.RND is set to 1 and PMSIDR_EL1.ERnd is 1:

  - PMSICR_EL1.COUNT[31:8] is set to PMSIRR_EL1.INTERVAL.
  - PMSICR_EL1.COUNT[7:0] is set to `0x00`.

3.1.2 Behavior of the sample interval counter while profiling is enabled

While profiling is enabled, the counters control when an operation is selected for sampling. The behavior depends on PMSIRR_EL1.RND and an IMPLEMENTATION DEFINED choice discoverable in PMSIDR_EL1.ERnd.

If PMSIRR_EL1.RND is cleared to 0:

While nonzero, the sample interval counter decrements by 1 for each member of the sample population. When the counter reaches zero:

- A member of the sampling population is selected for sampling.
- The counter is set as follows:
  - PMSICR_EL1.COUNT[31:8] is set to PMSIRR_EL1.INTERVAL.
  - PMSICR_EL1.COUNT[7:0] is set to `0x00`.

Note

Because the counter counts down to zero, when PMSIRR_EL1.RND is cleared to 0 the interval between operations being selected for sampling is $(INTERVAL \times 256 + 1)$.

If PMSIRR_EL1.RND is set to 1 and PMSIDR_EL1.ERnd is 0:

While nonzero, the sample interval counter decrements by 1 for each member of the sample population. When the counter reaches zero:

- A member of the sampling population is selected for sampling.
- The counter is set as follows:
  - PMSICR_EL1.COUNT[31:8] is set to PMSIRR_EL1.INTERVAL.
  - PMSICR_EL1.COUNT[7:0] is set to a random or pseudorandom value in the range `0x00` to `0xFF`.

Note

When PMSIRR_EL1.RND is set to 0 and PMSIDR_EL1.ERnd is 1, the mean interval between operations being selected for sampling is $(INTERVAL \times 256 + 128)$, if the random number generator is uniform.

If PMSIRR_EL1.RND is set to 1 and PMSIDR_EL1.ERnd is 1:

While nonzero, the primary sample interval counter decrements by 1 for each member of the sample population. When the primary counter reaches zero:

- The primary sample interval counter is reloaded.
- A secondary sample interval counter, PMSICR_EL1.ECOUNT, is set to a random or pseudorandom value in the range `0x00` to `0xFF`.

While the secondary sample interval counter is nonzero, the secondary sample interval counter decrements by 1 for each member of the sample population. The primary sample interval counter also continues to decrement because it is also nonzero.

When the secondary sample interval counter reaches zero, an operation is selected for sampling.

Note

When PMSIRR_EL1.RND is set to 1 and PMSIDR_EL1.ERnd is 1, the mean interval between operations being selected for sampling is $(INTERVAL \times 256 + 1)$, if the random number generator is uniform.

3.1.3 Behavior of the sample interval counter while profiling is disabled

When profiling is disabled:

- No operations are selected for sampling.
- No sample records are collected.
- The sample interval counters retain their values and do not decrement.

3.1.4 Sample collision behavior

On a sample collision:

- The PMU event SAMPLE_COLLISION is generated.
- PMBSR_EL1.COLL is set to 1.

Following a context synchronization event an indirect write to PMBSR_EL1.COLL is guaranteed to be visible to instructions in program order after the sampled operation that collided. There is no guarantee of visibility without a context synchronization event. For more information see Synchronization and Statistical Profiling.

Note

This means that following a context synchronization event PMBSR_EL1.COLL will not change on entry to a state where profiling is disabled.

3.2 Enable and Filtering controls

3.2.1 Enabling Profiling

Profiling is disabled if the Profiling Buffer is disabled, including when:

- PMBLIMITR_EL1.E is cleared to 0 or PMBSR_EL1.S is set to 1.
- Executing at a higher Exception level than the Profiling Buffer owning Exception level.
- Executing in the Security state that is not the Security state of the owning Exception level.
- The PE is in Debug state.

Note

The owning Exception level is controlled by MDCR_EL3.NSPB and MDCR_EL2.E2PB.

PMSCR_EL1.{E1SPE, E0SPE} and PMSCR_EL2.{E2SPE, E0HSPE} enable sampling by Exception level:

- In a guest operating system or Secure state, PMSCR_EL1.E1SPE enables profiling at EL1 and PMSCR_EL1.E0SPE at EL0.

- In a hypervisor or host operating system, PMSCR_EL2.E2SPE enables profiling at EL2 and PMSCR_EL2.E0HSPE at EL0.

Table 3 summarizes the controls for enabling by Exception level and Security state.

Table 3: Enabling by Exception level and Security state (for all Exception levels using AArch64)

Controls						**Sampling enabled at**			
S	**E**	**NS**	**NSPB**	**E2PB**	**TGE**	**EL3**	**EL2**	**EL1**	**EL0**
1	X	X	X	X	X	D	D	D	D
0	0	X	X	X	X	D	D	D	D
	1	1	`0b1X`	`0b1X`	0	D	D	E1SPE	E0SPE
					1	D	D	n/a	D
				`0b00`	0	D	E2SPE	E1SPE	E0SPE
					1	D	E2SPE	n/a	E0HSPE
			`0b0X`	X	X	D	D	D	D
		0	`0b1X`	X	X	D	n/a	D	D
			`0b0X`	X	X	D	n/a	E1SPE	E0SPE

S = PMBSR_EL1.S
E = PMBLIMITR_EL1.E
NS = SCR_EL3.NS
NSPB = MDCR_EL3.NSPB
E2PB = MDCR_EL2.E2PB
TGE = HCR_EL2.TGE

D = Disabled.
E2SPE = Enabled if PMSCR_EL2.E2SPE == 1, disabled otherwise.
E1SPE = Enabled if PMSCR_EL1.E1SPE == 1, disabled otherwise.
E0HSPE = Enabled if PMSCR_EL2.E0HSPE == 1, disabled otherwise.
E0SPE = Enabled if PMSCR_EL1.E0SPE == 1, disabled otherwise.

3.2.2 Filtering sample records

PMSFCR_EL1.FT enables filtering by operation type. When enabled PMSFCR_EL1.{ST, LD, B} define the collected types:

- ST enables collection of store sampled operations, including all atomic operations.
- LD enables collection of load sampled operations, including atomic operations that return a value to a register.
- B enables collection of branch sampled operations, including direct and indirect branches and exception returns.

Note

When µ-op sampling is implemented, filtering is based on the µ-op type.

Table 4 summarizes the controls for filtering by operation type.

Table 4: Filtering by Operation type

PMSFCR_EL1.				**Operation type**			
FT	**LD**	**ST**	**B**	**Load**	**Store**	**Branch**	**Other**
0	X	X	X	C	C	C	C
1	0	0	0	D	D	D	D
			1	D	D	C	D
		1	0	D	C	D	D
			1	D	C	C	D
	1	0	0	C	D	D	D
			1	C	D	C	D
		1	0	C	C	D	D
			1	C	C	C	D

D = Discarded.

C = Collected.

PMSFCR_EL1.FE enables filtering by a set of events that are defined by PMSEVFR_EL1. When enabled, only sampled operations with all the events in the filter set are recorded and written to the Profiling Buffer.

PMSFCR_EL1.FL enables filtering by total latency. PMSLATFR_EL1.MINLAT defines the minimum latency. When enabled, only sampled operations with a total latency greater than or equal to the minimum latency are recorded and written to the Profiling Buffer.

These controls combine together as a logical AND.

Example

If PMSFCR_EL1.FE is set to 1, PMSFCR_EL1.FT is set to 1, and PMSFCR_EL1.FL is set to 1, then only sampled operations that meet all of the following criteria are recorded and written to the Profiling Buffer:

- The sampled operation is one of the selected operation types.
- The operation has all of the events in the filter set.
- The total latency is equal to or greater than the minimum latency.

3.2.3 Pseudocode details of filtering and enable controls

```
// StatisticalProfilingEnabled()
// =============================
boolean StatisticalProfilingEnabled()
    if !HasStatisticalProfiling() || UsingAArch32() || !ProfilingBufferEnabled() then
        return FALSE;

    in_host = !IsSecure() && HaveEL(EL2) && HCR_EL2.TGE == '1';
    (secure, el) = ProfilingBufferOwner();
    if UInt(el) < UInt(PSTATE.EL) || secure != IsSecure() || (in_host && el == EL1) then
        return FALSE;

    case PSTATE.EL of
        when EL3  Unreachable();
        when EL2  spe_bit = PMSCR_EL2.E2SPE;
        when EL1  spe_bit = PMSCR_EL1.E1SPE;
        when EL0  spe_bit = (if in_host then PMSCR_EL2.E0HSPE else PMSCR_EL1.E0SPE);

    return spe_bit == '1';

enumeration SysRegAccess { SysRegAccess_OK, SysRegAccess_UNDEFINED, SysRegAccess_TrapToEL1,
SysRegAccess_TrapToEL2, SysRegAccess_TrapToEL3 };

// CheckStatisticalProfilingAccess()
// =================================
enumeration SysRegAccess CheckStatisticalProfilingAccess()
    if !HasStatisticalProfiling() || PSTATE.EL == EL0 || UsingAArch32() then
        return SysRegAccess_UNDEFINED;
    if HaveEL(EL2) && !IsSecure() && PSTATE.EL == EL1 && MDCR_EL2.TPMS == '1' then
        return SysRegAccess_TrapToEL2;
    if HaveEL(EL3) && PSTATE.EL != EL3 && MDCR_EL3.NSPB != SCR_EL3.NS:'1' then
        return SysRegAccess_TrapToEL3;
    return SysRegAccess_OK;

enumeration OpType { OpType_Other, OpType_Load, OpType_Store, OpType_Branch };

// CollectRecord()
// ===============
boolean CollectRecord(bits(64) events, integer total_latency, OpType optype)
    assert StatisticalProfilingEnabled();
    if PMSFCR_EL1.FE == '1' then
        e = events<63:48,31:24,15:12,7,5,3,1>;
        m = PMSEVFR_EL1<63:48,31:24,15:12,7,5,3,1>;
        // Check for UNPREDICTABLE case
        if IsZero(PMSEVFR_EL1) && ConstrainedUnpredictableBool() then return FALSE;
        if !IsZero(NOT(e) AND m) then return FALSE;
    if PMSFCR_EL1.FT == '1' then
        // Check for UNPREDICTABLE case
        if IsZero(PMSFCR_EL1.<B,LD,ST>) && ConstrainedUnpredictableBool() then
            return FALSE;
        case optype of
            when OpType_Branch  if PMSFCR_EL1.B  == '0' then return FALSE;
            when OpType_Load    if PMSFCR_EL1.LD == '0' then return FALSE;
            when OpType_Store   if PMSFCR_EL1.ST == '0' then return FALSE;
            otherwise           return FALSE;
    if PMSFCR_EL1.FL == '1' then
        if IsZero(PMSLATFR_EL1.MINLAT) && ConstrainedUnpredictableBool() then  // UNPREDICTABLE case
            return FALSE;
        if total_latency < UInt(PMSLATFR_EL1.MINLAT) then return FALSE;
    return TRUE;
```

3.3 Controlling the data that is collected

Certain data in sample records is only collected if permitted by one or both of EL1 and EL2. This is to restrict exposure of data to a lower Exception level or to Non-secure state.

CONTEXTIDR_EL1 is collected only if PMSCR_EL1.CX is set to 1 and one of the following is true:

- The PE is executing at EL1 or Secure EL0.
- The PE is executing at Non-secure EL0 and EL2 is not implemented.
- The PE is executing at Non-secure EL0 and HCR_EL2.TGE is cleared to 0.

CONTEXTIDR_EL2 is collected only if EL2 is implemented, PMSCR_EL2.CX is set to 1 and the PE is in the Non-secure state.

Timestamps are collected only if one of the following is true:

- PMSCR_EL1.TS is set to 1 and the Profiling Buffer is owned by EL1.
- PMSCR_EL2.TS is set to 1 and the Profiling Buffer is owned by EL2.

The timestamp is a choice between:

- Physical time, which is defined by the value of CNTPCT_EL0.
- Virtual time, as defined by the value of CNTVCT_EL0. That is, the physical time minus the virtual offset, CNTVOFF_EL2. However, the virtual offset is treated as zero if a read of CNTVCT_EL0 at the current Exception level would treat the virtual offset as zero.

Physical time is collected if any of the following is true:

- PMSCR_EL1.PCT is set to 1 and the Profiling Buffer is owned by Secure EL1.
- PMSCR_EL2.PCT is set to 1 and the Profiling Buffer is owned by EL2.
- PMSCR_EL1.PCT is set to 1 and PMSCR_EL2.PCT is set to 1 and the Profiling Buffer is owned by Non-secure EL1.

Virtual time is collected otherwise. If EL2 is not implemented, PMSCR_EL1.PCT is RES1.

Physical data addresses are collected only if one of the following is true:

- PMSCR_EL1.PA is set to 1 and the Profiling Buffer is owned by Secure EL1.
- PMSCR_EL2.PA is set to 1 and the Profiling Buffer is owned by EL2.
- PMSCR_EL1.PA is set to 1 and PMSCR_EL2.PA is set to 1 and the Profiling Buffer is owned by Non-secure EL1.

If EL2 is not implemented, the PE behaves as if PMSCR_EL2.PA is set to 1, other than for a direct read of the register.

Enabling collection of the physical data addresses has an IMPLEMENTATION DEFINED impact on the sampled operation.

See The owning Exception level.

3.3.1 Pseudocode details of collection controls

```
// CollectContextIDR1()
// ====================
boolean CollectContextIDR1()
    if !StatisticalProfilingEnabled() then return FALSE;
    if PSTATE.EL == EL2 then return FALSE;
    if HaveEL(EL2) && !IsSecure() && HCR_EL2.TGE == '1' then return FALSE;
    return PMSCR_EL1.CX == '1';


// CollectContextIDR2()
// ====================
boolean CollectContextIDR2()
    if !StatisticalProfilingEnabled() then return FALSE;
    if !HaveEL(EL2) || IsSecure() then return FALSE;
    return PMSCR_EL2.CX == '1';

enumeration TimeStamp { TimeStamp_None, TimeStamp_Virtual, TimeStamp_Physical };

// CollectTimeStamp()
// ==================
TimeStamp CollectTimeStamp()
    if !StatisticalProfilingEnabled() then return FALSE;
   (secure, el) = ProfilingBufferOwner();
   if el == EL2 then
       if PMSCR_EL2.TS == '0' then return TimeStamp_None;
   else
       if PMSCR_EL1.TS == '0' then return TimeStamp_None;
   if !secure && HaveEL(EL2) then
       pct = PMSCR_EL2.PCT == '1' && (el == EL2 || PMSCR_EL1.PCT == '1');
   else
       pct = PMSCR_EL1.PCT == '1';
   return (if pct then TimeStamp_Physical else TimeStamp_Virtual);

// CollectPhysicalAddress()
// ========================
boolean CollectPhysicalAddress()
    if !StatisticalProfilingEnabled() then return FALSE;
   (secure, el) = ProfilingBufferOwner();
   if !secure && HaveEL(EL2) then
       return PMSCR_EL2.PA == '1' && (el == EL2 || PMSCR_EL1.PA == '1');
   else
       return PMSCR_EL1.PA == '1';
```

3.4 The Profiling Buffer

The profile data is collected in a memory *Profiling Buffer*. The Profiling Buffer is defined by:

- PMBPTR_EL1, the current write pointer.
- PMBLIMITR_EL1, the write limit pointer.

The Profiling Buffer starts at the current write pointer and extends to the current limit pointer minus one. The write limit pointer must be aligned to the smallest implemented translation granule size. The alignment of the current write pointer is IMPLEMENTATION DEFINED.

PMBLIMITR_EL1 and PMBPTR_EL1 are virtual addresses in the stage 1 translation regime of the owning Exception level. This is called the *owning translation regime*.

Note

The translation of virtual addresses to physical addresses is identical to that for any other virtual address in the owning Exception level. For example, PMBPTR_EL1[63:56] are ignored by address translation if the respective TBI bit is set to 1.

3.4.1 Restrictions on the current write pointer

This section describes the software rules on setting the current write pointer, PMBPTR_EL1. If these rules are not followed, the behavior is UNPREDICTABLE and the PE might do any of the following at any point after profiling is enabled:

- Write sample records to any writeable address in memory.
- Generate a Profiling Buffer management event indicating the Profiling Buffer is full with or without data loss.
- Generate a Profiling Buffer management event indicating a Translation Fault with or without data loss.

When profiling becomes enabled, all the following must be true:

- The current write pointer must be at least one sample record below the write limit pointer. That is:

  `UInt`(PMBPTR_EL1.PTR) <= `UInt`(PMBLIMITR_EL1.LIMIT:`Zeros`(12)) - $2^{\text{PMSIDR\_EL1.MaxSize}}$.
- PMBPTR_EL1.PTR[63:56] must equal PMBLIMITR_EL1.LIMIT[63:56].

When the Profiling Buffer is first configured, PMBPTR_EL1.PTR must be aligned to $2^{\text{PMBIDR\_EL1.Align}}$. That is, if PMBIDR_EL1.Align is nonzero, PMBPTR_EL1.PTR [`UInt`(PMBIDR_EL1.Align)-1:0] must be all zeros.

However, the current write pointer can usually be restored to the saved write pointer value it had when profiling was disabled, providing a `PSB CSYNC` and a context synchronization event were executed before reading PMBPTR_EL1:

- If no Profiling Buffer management event was signaled then profiling can be restarted from the saved write pointer. In this case, the saved write pointer points within one sample record of the write limit pointer.
- If a Profiling Buffer management event was signaled then:
  - If PMBSR_EL1.S is restored to 1, then profiling is not being enabled, and there are no constraints on the value written to PMBPTR_EL1.
  - If PMBSR_EL1.S is restored to 0, and the Profiling Buffer management event was caused by an MMU fault, profiling can be restarted from the saved write pointer, if the following is true.
    - PMBSR_EL1.{EA, DL} did not also indicate an external abort or data loss.
    - The saved write pointer is at least one sample record below the write limit pointer.

Note

If a signaled MMU fault has not been corrected, the Statistical Profiling Extension generates a new MMU fault Profiling Buffer management event when it next tries to write a sample record.

- If PMBSR_EL1.S is restored to 0, and the Profiling Buffer management event was caused by a buffer full event, the Profiling Buffer can be extended and profiling restarted from the saved write pointer if the following is true.
  - PMBSR_EL1.{EA, DL} did not also indicate an external abort or data loss.
  - The saved write pointer is at least one sample record below the extended write limit pointer.

The current write pointer must not be restored from the saved write pointer following a Profiling Buffer management event if PMBSR_EL1.DL was set to 1.

The saved write pointer might not be aligned to $2^{\text{PMBIDR\_EL1.Align}}$ and might point to within one sample record of the write limit pointer.

For more information see Synchronization and Statistical Profiling.

3.4.2 The owning Exception level

The owning Exception level is:

- Non-secure EL1, if all of the following are true:
  - Either EL3 is not implemented and the PE executes in Non-secure state, or MDCR_EL3.NSPB is set to either `0b10` or `0b11`.
  - Either EL2 is not implemented or MDCR_EL2.E2PB is set to either `0b10` or `0b11`.
- EL2, if all of the following are true:
  - EL2 is implemented.
  - Either EL3 is not implemented or MDCR_EL3.NSPB is set to either `0b10` or `0b11`.
  - MDCR_EL2.E2PB is set to `0b00`.
- Secure EL1, if any of the following are true:
  - EL3 is not implemented and the PE executes in Secure state.
  - MDCR_EL3.NSPB is set to either `0b00` or `0b01`.

When the owning Exception level is Non-secure EL1:

The Profiling Buffer addresses are in the Non-secure EL1&0 translation regime using the current ASID from TTBRx_EL1. This is a two-stage translation using the current VMID if EL2 is implemented and HCR_EL2.VM is set to 1.

If EL3 is implemented, then the Profiling Buffer is disabled in Secure state.

If EL2 is implemented, then profiling is disabled at EL2 and at Non-secure EL0 when HCR_EL2.TGE is set to 1.

When the owning Exception level is EL2:

The Profiling Buffer addresses are in the EL2 translation regime. If both HCR_EL2.E2H is set to 1 and HCR_EL2.TGE is set to 1, this is an EL2&0 translation regime using the current EL2&0 translation regime ASID from TTBRx_EL2.

If EL3 is implemented, then the Profiling Buffer is disabled in Secure state.

Note

If either HCR_EL2.E2H is cleared to 0 or HCR_EL2.TGE is cleared to 0, and the PE is executing at EL1 or EL0, the EL2 translation regime is not the current stage 1 translation regime because the current stage 1 translation regime is EL1&0.

When the owning Exception level is Secure EL1:

The Profiling Buffer addresses are in the Secure EL1&0 translation regime using the current ASID from TTBRx_EL1.

If EL3 is implemented, then the buffer is disabled in Non-secure state.

The Profiling Buffer is disabled if any of the following are true:

- SCR_EL3.NS indicates the other Security state to the owning Exception level.
- The owning Exception level is using AArch32 state.
- PMBLIMITR_EL1.E is cleared to 0.

Table 5 summarizes the owning translation regime.

Table 5: Summary of owning translation regime (for all Exception levels using AArch64 state)

PMBLIMITR_EL1.E	SCR_EL3.NS	MDCR_EL3.NSPB	MDCR_EL2.E2PB	Owning translation regime
0	X	X	X	Disabled
1	1	`0b1X`	`0b1X`	Non-secure EL1&0
			`0b00`	Non-secure EL2 or EL2&0[a]
		`0b0X`	X	Disabled
	0	`0b1X`	X	Disabled
		`0b0X`	X	Secure EL1&0

a. Depending on the values of HCR_EL2.{E2H,TGE}.

3.4.3 Memory access types and coherency

The Statistical Profiling Extension acts as a separate observer in the system and is subject to the rules regarding coherency. For more information, see the *ARM® Architecture Reference Manual, ARMv8, for ARMv8-A architecture profile*.

The memory type that is used for a write by the Statistical Profiling Extension to the Profiling Buffer is taken from the translation table entries for the virtual address being written to. That is:

- The writes are treated as coming from an observer that is coherent with all observers in the Shareability domain that is defined by the translation tables.
- There is no requirement to manage coherency for observers in the same Shareability domain but coherency for other observers in the system might require explicit management.

For more information see Synchronization and Statistical Profiling.

3.4.4 TLB operations

Translations might be cached in a TLB. TLB maintenance operations (including TLB maintenance operations that are broadcast from other PEs) operate on these cached translations.

Any TLB maintenance operation followed by a `DSB` (including an operation broadcast from another PE and a following `DSB` on that other PE):

- Does not complete until all sample records previously translated using an invalidated translation have been written to the Profiling Buffer.
- Any remaining sample records are guaranteed not to use an invalidated translation.

For such operations, the `DSB` must apply to both loads and stores. Although the Statistical Profiling Extension acts as another observer in the system, for determining the Shareability domain of this `DSB` or TLB maintenance operation, the writes of sample records are treated as coming from the PE that is being profiled.

Note

Completion of the `DSB` does not guarantee that all buffered profiling data has written out. See Synchronization and Statistical Profiling.

3.4.5 Effect on the exclusive monitors

If an operation between Load-Exclusive and Store-Exclusive instructions is sampled, then the Store-Exclusive must be guaranteed not to fail, even though the sample record is written to an unrelated address.

3.4.6 Pseudocode details of the Profiling Buffer

```
// ProfilingBufferOwner()
// ======================
(boolean, bits(2)) ProfilingBufferOwner()
    secure = if HaveEL(EL3) then (MDCR_EL3.NSPB<1> == '0') else IsSecure();
    el = if !secure && HaveEL(EL2) && MDCR_EL2.E2PB == '00' then EL2 else EL1;
    return (secure, el);


// ProfilingBufferEnabled()
// ========================
boolean ProfilingBufferEnabled()
    if !HasStatisticalProfiling() then return FALSE;
    (secure, el) = ProfilingBufferOwner();
    non_secure_bit = if secure then '0' else '1';
    return (!ELUsingAArch32(el) && non_secure_bit == SCR_EL3.NS &&
            PMBLIMITR_EL1.E == '1' && PMBSR_EL1.S == '0');


// CheckProfilingBufferAccess()
// ============================
enumeration SysRegAccess CheckProfilingBufferAccess()
    if !HasStatisticalProfiling() || PSTATE.EL == EL0 || UsingAArch32() then
        return SysRegAccess_UNDEFINED;

    if HaveEL(EL2) && !IsSecure() && PSTATE.EL == EL1 && MDCR_EL2.E2PB<0> != '1' then
        return SysRegAccess_TrapToEL2;

    if HaveEL(EL3) && PSTATE.EL != EL3 && MDCR_EL3.NSPB != SCR_EL3.NS:'1' then
        return SysRegAccess_TrapToEL3;

    return SysRegAccess_OK;
```

3.5 Profiling Buffer management event

A Profiling Buffer management event occurs:

- On a fault, see Faults and watchpoints.
- On an external abort, see External aborts.
- When the Profiling Buffer fills, see Buffer full event.

On a Profiling Buffer management event:

- The service bit, PMBSR_EL1.S, is set to 1.
- The data loss bit, PMBSR_EL1.DL, is set as described in the event description.
- The Profiling Buffer management interrupt signal, **PMBIRQ**, is asserted:
  - **PMBIRQ** is a level interrupt driven by PMBSR_EL1.S. This means that:
    - A direct write that sets PMBSR_EL1.S to 1 causes the interrupt to be asserted.
    - **PMBIRQ** remains asserted until software clears PMBSR_EL1.S to 0.
  - If a *Generic Interrupt Controller* (GIC) is implemented, **PMBIRQ** must be configured as a *Private Peripheral Interrupt* (PPI) in a multiprocessor system. **PMBIRQ** is signaled by the PE that implements the Statistical Profiling Extension.

Note

A standard PPI number is allocated by the *ARM® Server Base System Architecture* (SBSA).

- Additional syndrome for the event is written to PMBSR_EL1.MSS. Unless otherwise stated in the event description, other PMBSR_EL1 fields are unchanged.

While PMBSR_EL1.S is set to 1:

- The buffer is disabled and profiling is disabled.
- All remaining buffered sample records are discarded.
- The values in PMBPTR_EL1 are retained and PMSICR_EL1 does not decrement.

Buffer full events and MMU faults leading to Profiling Buffer management events are synchronous.

It is IMPLEMENTATION DEFINED whether external aborts are reported to the Statistical Profiling Extension synchronously or asynchronously. If external aborts are reported as asynchronous:

- The external abort might not be received until after a first Profiling Buffer management event has set PMBSR_EL1.S to 1.
- Writes to the buffer might generate a second Profiling Buffer management event after the external abort has set PMBSR_EL1.S to 1.

The architecture does not require that a sample record is written sequentially by the Statistical Profiling Extension, only that:

- The Statistical Profiling Extension never writes past the PMBLIMITR_EL1 limit pointer.
- On a Profiling Buffer management event, PMBSR_EL1.DL indicates whether PMBPTR_EL1 points to the first byte after the last complete sample record.
- On an MMU fault or synchronous external abort, PMBPTR_EL1 serves as a Fault Address Register.

Note

This means that software must not assume that:

- There is ever any valid data beyond the current PMBPTR_EL1 write pointer.
- The PE has not written a valid sample record between the current PMBPTR_EL1 write pointer and the PMBLIMITR_EL1 limit pointer.
- If PMBSR_EL1.DL is set to 1 on a Profiling Buffer management event, that there is any valid data between the end of the last complete sample record and the current PMBPTR_EL1 write pointer.
- *Any* valid data has been written to the Profiling Buffer if an external abort is reported asynchronously to the Statistical Profiling Extension.

The last complete sample record must end at most $2^{(\text{PMSIDR\_EL1.MaxSize})}$ bytes below PMBPTR_EL1.

3.5.1 Prioritization of Profiling Buffer management events

Where multiple synchronous Profiling Buffer management events occur on writing a sample record, the PE prioritizes them as follows (from highest to lowest priority):

1. Synchronous fault.
2. Synchronous external abort.

3. Buffer full event.

Asynchronous external aborts are not prioritized with respect to other events.

3.5.2 Buffer full event

If, after writing a sample record, there is not sufficient space in the Profiling Buffer for a sample record of the size indicated by PMSIDR_EL1.MaxSize, and PMBSR_EL1.S is 0, a Profiling Buffer management event is generated:

- PMBSR_EL1.EC is set to `0b000000`, other buffer management event.
- The BSC field of PMBSR_EL1.MSS is set as follows:
  - PMBSR_EL1.BSC is set to `0b000001`, buffer filled.
- PMBPTR_EL1 is set to the first byte after the last complete sample record. PMBSR_EL1.DL is unchanged.
- The other PMBSR_EL1 fields are unchanged.

That is, the Profiling Buffer management event is generated when the PE writes past the write limit pointer minus $2^{(\text{PMSIDR\_EL1.MaxSize})}$. The Statistical Profiling Extension never writes beyond the write limit pointer.

For more information see Restrictions on the current write pointer.

3.5.3 Faults and watchpoints

Table 6 lists the faults that might be generated by a write to the Profiling Buffer by the Statistical Profiling Extension.

Writes to the Profiling Buffer never generate watchpoints.

Table 6: Summary of faults

Fault	Conditions
Translation	Any access outside the virtual address space generates a Translation fault.
	The translation of a virtual address to a physical address might generate a Translation fault.
Address Size	The translation of a virtual address to a physical address might generate an Address Size fault.
Alignment	If PMBPTR_EL1 is not aligned to an IMPLEMENTATION DEFINED minimum alignment, writes to the Profiling Buffer generate an Alignment fault.
Permission	Writes to the Profiling Buffer are made as privileged writes. If the write does not have write permission, a Permission fault is generated. The value of PSTATE.PAN is ignored and treated as zero.
Unsupported Access	If hardware management of the Dirty Bit is enabled and the Dirty Bit is not set, it is IMPLEMENTATION DEFINED whether an Unsupported Access fault is generated or the write sets the Dirty Bit in the translation table.
Access Flag	If hardware management of the Access Flag is enabled and the Access Flag is not set, it is IMPLEMENTATION DEFINED whether an Access Flag fault is generated or the write sets the Access Flag in the translation table.

Fault	**Conditions**
TLB Conflict fault	IMPLEMENTATION DEFINED.

If a write to the Profiling Buffer generates a fault and PMBSR_EL1.S is 0, then a Profiling Buffer management event is generated:

- PMBSR_EL1.S is set to 1.
- PMBSR_EL1.EC is set to one of:
  - `0b100100`, stage 1 Data Abort on write to the Profiling Buffer.
  - `0b100101`, stage 2 Data Abort on write to the Profiling Buffer.
- The FSC field of PMBSR_EL1.MSS is set as follows:
  - PMBSR_EL1.FSC is set to indicate the type of the fault.
- PMBPTR_EL1 is set to the address that generated the fault.
- If PMBPTR_EL1 is not the address of the first byte after the last complete sample record written by the Statistical Profiling Extension, then PMBSR_EL1.DL is set to 1. Otherwise, PMBSR_EL1.DL is unchanged.
- The other PMBSR_EL1 fields are unchanged.

Note

Each of these faults gives rise to a Profiling Buffer management event, not an actual MMU fault exception. The ESR and FAR registers are unchanged.

3.5.4 External aborts

A write to the Profiling Buffer might generate an external abort, including an external abort on a translation table walk or translation table update. It is an IMPLEMENTATION DEFINED choice whether such an external abort:

- Is reported to the Statistical Profiling Extension and treated as a Profiling Buffer management event.
- Generates an SError interrupt exception.

If a write to the Profiling Buffer generates an external abort that is reported to the Statistical Profiling Extension:

- The external abort bit, PMBSR_EL1.EA, is set to 1.
- The Statistical Profiling Extension stops writing sample records to the Profiling Buffer. It is IMPLEMENTATION DEFINED whether an external abort on a write to the Profiling Buffer is reported as synchronous or asynchronous:
  - The external abort is reported as *synchronous* if PMBPTR_EL1 is set to the address that was externally aborted.
  - The external abort is reported as *asynchronous* if PMBPTR_EL1 is not guaranteed to be set to the address that was externally aborted.

- If PMBSR_EL1.S is zero, a Profiling Buffer management event is generated:
  - PMBSR_EL1.S is set to 1.
  - PMBSR_EL1.EC is set to one of:
    - `0b100100`, stage 1 Data Abort on write to Profiling Buffer.
    - `0b100101`, stage 2 Data Abort on write to Profiling Buffer.
  - The FSC field of PMBSR_EL1.MSS is set for a stage 1 or stage 2 Data Abort as follows:
    - PMBSR_EL1.FSC is set to indicate the type of external abort.
- If the external abort is reported as asynchronous or PMBPTR_EL1 is not the address of the first byte of the sample record being written by the Statistical Profiling Extension, then PMBSR_EL1.DL is set to 1. Otherwise PMBSR_EL1.DL is unchanged.

  Note

  Following an external abort reported asynchronously to the Statistical Profiling Extension, software must not assume that *any* valid data has been written to the Profiling Buffer.

- The other PMBSR_EL1 fields are unchanged.

If a write to the Profiling Buffer generates an external abort that is taken as an SError interrupt exception, the PE takes the SError interrupt exception as normal, and PMBSR_EL1 fields are unchanged.

Note

Treating the external abort as a Profiling Buffer management event:

- Sets PMBSR_EL1.S to 1 and so disables the Statistical Profiling Extension.
- Allows error recovery software to isolate the event to the actions of the Statistical Profiling Extension.

Taking an SError interrupt:

- Means that the Statistical Profiling Extension will only be disabled if the SError interrupt is taken to an Exception level where the Statistical Profiling Extension is disabled.
- Might not allow error recovery software to isolate the event and error containment.

3.6 Synchronization and Statistical Profiling

The profiling operation of the Statistical Profiling Extension:

- Makes indirect reads and indirect writes of System registers.
- Writes to memory.
- Makes further indirect writes to PMBPTR_EL1 as a result of an external abort on a write to memory.

The indirect reads of the PMSCR_EL1.{E1SPE, E0SPE} and PMSCR_EL2.{E2SPE, E0HSPE} controls when determining whether to select an operation for profiling are treated as indirect reads made by the instruction being executed, and subject to the standard requirements for synchronization.

Otherwise, although the profiling operation is generated by a sampled operation, the profiling operation executes independently of the instructions that are executed on the PE, and acts as a separate memory observer from the PE in the system.

A `DSB` instruction guarantees that all memory transactions that are made by the PE are observable by writes made by a profiling operation relating to a sampled operation in program order after the `DSB` instruction.

A context synchronization event guarantees that a direct write to a System register made by the PE in program order before the context synchronization event are observable by indirect reads and indirect writes of the same System register made by a profiling operation relating to a sampled operation in program order after the context synchronization event.

To synchronize previous profiling operations, software must execute a `PSB CSYNC` Buffer Synchronization instruction.

—— Note ——————————

The `PSB CSYNC` instruction is not defined in the AArch32 instruction set architecture.

——————————————

Following a context synchronization event, a `PSB CSYNC` instruction is guaranteed to synchronize the profiling operations for all instructions that are executed in program order before the context synchronization event.

Synchronized by the `PSB CSYNC` instruction means:

- A direct read of a System register in program order following a `PSB CSYNC` instruction requires explicit synchronization to observe an indirect write to the same System register made by a profiling operation synchronized by the `PSB CSYNC` instruction.
- An indirect write to a System register made by a profiling operation synchronized by a `PSB CSYNC` instruction does not affect a direct write to the same System register made in program order following the `PSB CSYNC` instruction.
- A direct write to a System register in program order following a `PSB CSYNC` instruction is not allowed to affect an indirect read of the same System register made by a profiling operation synchronized by the `PSB CSYNC` instruction.
- A `DSB` instruction in program order following a `PSB CSYNC` instruction does not complete before the writes to the Profiling Buffer of sample records for profiling operations synchronized by the `PSB CSYNC` instruction have completed. The `DSB` instruction must apply to both loads and stores.

For the indirect write to PMBSR_EL1 that is made as a result of an external abort on a write of a sample record to memory, the synchronization rules apply only after the write has completed.

Although the Statistical Profiling Extension acts as another observer in the system, for determining the Shareability domain of the `DSB` instructions, the writes of sample records are treated as coming from the PE that is being profiled.

Note

If the Statistical Profiling Extension is not disabled when the context synchronization event occurs, further profiling operations might be generated that are not guaranteed to be synchronized by the `PSB CSYNC` instruction.

If the PE takes an exception to an Exception level where the Statistical Profiling Extension is disabled, no new operations are selected for sampling. The Statistical Profiling Extension is always disabled if the owning Exception level is a lower Exception level than the current Exception level.

In the absence of a context synchronization event, a `PSB CSYNC` instruction is not required to execute in program order with respect to sampled operations.

3.6.1 UNPREDICTABLE behavior

In the absence of correct context synchronization events, it is UNPREDICTABLE whether an indirect read of a System register made by a profiling operation will return the old or the new values.

If the indirect reads mean that `ProfilingBufferEnabled()` returns FALSE when a sample record or records are about to be written to the physical address, then it is further UNPREDICTABLE whether the sample record or records:

- Are written to memory.
- Are silently discarded and not written to memory.
- Are discarded and not written to memory, and a Profiling Buffer management event is generated:
  - PMBSR_EL1.DL, is set to 1.
  - PMBSR_EL1.EC is set to `0x00`.
  - PMBSR_EL1.BSC is set to `0x00` to indicate that the buffer is not full.

This means that software must execute a `PSB CSYNC` instruction to force any sample records to be written to the Profiling Buffer before changing context.

4 Detailed Programmers Model

4.1 Register index

Table 7: AArch64 System register map

op0	op1	CRn	CRm	op2	Access	Register	Description
3	0	c9	c9	0	R/W	PMSCR_EL1	Statistical Profiling Control Register (EL1)
				2	R/W	PMSICR_EL1	Sampling Interval Counter Register
				3	R/W	PMSIRR_EL1	Sampling Interval Reload Register
				4	R/W	PMSFCR_EL1	Sampling Filter Control Register
				5	R/W	PMSEVFR_EL1	Sampling Event Filter Register
				6	R/W	PMSLATFR_EL1	Sampling Latency Filter Register
				7	RO	PMSIDR_EL1	Sampling Profiling ID Register
			c10	0	R/W	PMBLIMITR_EL1	Profiling Buffer Limit Address Register
				1	R/W	PMBPTR_EL1	Profiling Buffer Write Pointer Register
				3	R/W	PMBSR_EL1	Profiling Buffer Status/syndrome Register
				7	RO	PMBIDR_EL1	Profiling Buffer ID Register
	4	c9	c9	0	R/W	PMSCR_EL2	Statistical Profiling Control Register (EL2)
	5	c9	c9	0	R/W	PMSCR_EL12	Statistical Profiling Control Register (EL1)

4.2 Register resets

Table 8: Reset values

Register	Field	Reset value
PMBLIMITR_EL1	E	0

All other resettable registers and fields are reset to an IMPLEMENTATION DEFINED value, which might be UNKNOWN.

4.3 Register descriptions

4.3.1 PMBIDR_EL1, Profiling Buffer ID Register

The PMBIDR_EL1 characteristics are:

Purpose

Provides information to software as to whether the buffer can be programmed at the current Exception level.

Usage constraints

Direct reads of PMBIDR_EL1 are UNDEFINED at EL0.

Traps and enables

None.

Configurations

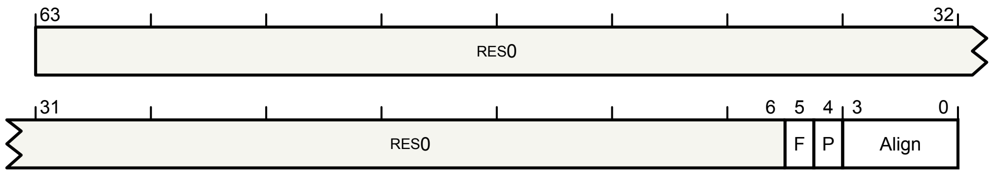
Present only if the Statistical Profiling Extension is implemented. Direct reads of PMBIDR_EL1 are UNDEFINED otherwise.

Attributes

PMBIDR_EL1 is a 64-bit read-only AArch64 System register accessed using `MRS` of `S3_0_c9_c10_7`.

Field descriptions

The PMBIDR_EL1 bit assignments are:



Bits [63:6]

Reserved. This field is RES0.

F, bit [5]

Flag Updates. Defines whether the address translation performed by the Profiling Buffer manages the Access Flag and Dirty Bit. The defined values of this bit are:

`0` Accesses to pages not marked with Access Flag and not with Dirty Bit set will generate an Unsupported Access fault if hardware management of those flags is enabled.

`1` Profiling Buffer address translation manages the Access Flag and Dirty Bit in the same way as the MMU on this PE.

P, bit [4]

Prohibited. The Profiling Buffer is owned by the current or a lower Exception level in the current security state. The defined values of this bit are:

`0` Profiling Buffer is owned by the current or a lower Exception level in the current security state.

This does not mean an access will not be trapped to a higher Exception level.

`1` Profiling Buffer is owned by a higher Exception level or the other security state.

The value read from this field depends on the current Exception level and the values of MDCR_EL3.NSPB and MDCR_EL2.E2PB:

- If MDCR_EL3.NSPB == `0b10` or `0b11`, this bit reads as one from Secure EL1.
- If MDCR_EL2.E2PB == `0b00`, this bit reads as one from Non-secure EL1.
- If MDCR_EL3.NSPB == `0b00` or `0b01` this bit reads as one from EL2 and Non-secure EL1.
- Otherwise, this bit reads as zero.

Align, bits [3:0]

Defines the minimum alignment constraint for PMBPTR_EL1. If this field is non-zero, then the PE must pad every record up to a multiple of this size. The defined values of this field are:

`0b0000`	Byte.
`0b0001`	Halfword. PMBPTR_EL1[0] is RES0.
`0b0010`	Word. PMBPTR_EL1[1:0] is RES0.
`0b0011`	Doubleword. PMBPTR_EL1[2:0] is RES0.
...	...
`0b1011`	2KB. PMBPTR_EL1[10:0] is RES0.

All other values are reserved. Reserved values might be defined in a future version of the architecture.

4.3.2 PMBLIMITR_EL1, Profiling Buffer Limit Address Register

The PMBLIMITR_EL1 characteristics are:

Purpose

Defines the upper limit for the profiling buffer, and enables the profiling buffer.

Usage constraints

Direct reads and writes of PMBLIMITR_EL1 are UNDEFINED at EL0.

Traps and enables

Subject to the exception prioritization rules:

- If MDCR_EL2.E2PB == `0b00` or `0b10`, then direct reads and writes of PMBLIMITR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
- If MDCR_EL3.NSPB ≠ `0b01`, then direct reads and writes of PMBLIMITR_EL1 at Secure EL1 generate a Trap exception to EL3.
- If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMBLIMITR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

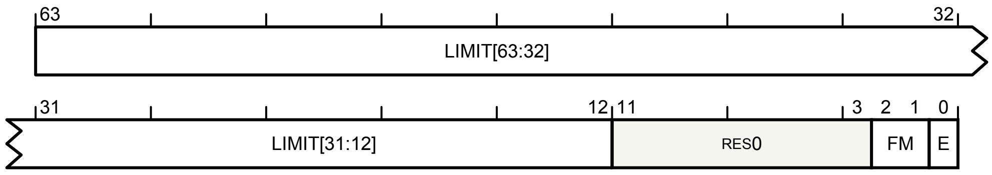
Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMBLIMITR_EL1 are UNDEFINED otherwise.

Attributes

PMBLIMITR_EL1 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_0_c9_c10_0`.

Field descriptions

The PMBLIMITR_EL1 bit assignments are:



LIMIT[63:12], bits [63:12]

Limit address, plus one. Read/write. Defines the limit of the buffer. If the smallest implemented translation granule is not 4KB, then bits [N-1:12] are RES0, where N is the IMPLEMENTATION DEFINED value, Log$_2$(smallest implemented translation granule).

Bits [11:3]

Reserved. This field is RES0.

FM, bits [2:1]

Fill mode. The possible values of this field are:

`0b00`	Stop collection and raise maintenance interrupt on buffer fill.

All other values are reserved. If this field is programmed with a reserved value, the PE behaves as if this field has a defined value, other than for a direct read of the register. Software must not rely on the behavior of reserved values, as they might change in a future version of the architecture.

E, bit [0]

Profiling Buffer enable. The possible values of this bit are:

`0`	All output is discarded.
`1`	Enabled.

This bit resets to zero.

4.3.3 PMBPTR_EL1, Profiling Buffer Write Pointer Register

The PMBPTR_EL1 characteristics are:

Purpose

Defines the current write pointer for the profiling buffer.

Usage constraints

Direct reads and writes of PMBPTR_EL1 are UNDEFINED at EL0.

Traps and enables

Subject to the exception prioritization rules:

- If MDCR_EL2.E2PB == `0b00` or `0b10`, then direct reads and writes of PMBPTR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
- If MDCR_EL3.NSPB ≠ `0b01`, then direct reads and writes of PMBPTR_EL1 at Secure EL1 generate a Trap exception to EL3.
- If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMBPTR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

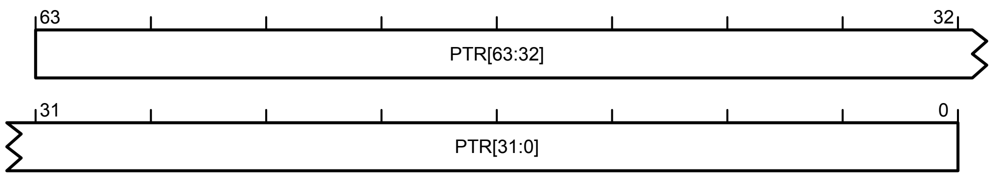
Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMBPTR_EL1 are UNDEFINED otherwise.

Attributes

PMBPTR_EL1 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_0_c9_c10_1`.

Field descriptions

The PMBPTR_EL1 bit assignments are:



PTR, bits [63:0]

Current write address. Defines the virtual address of the next entry to be written to the buffer.

Software must treat bits [M:0] of this register as RES0, where M is defined by PMBIDR_EL1.Align. If synchronous reporting of external aborts is not supported, then hardware also treats these bits as RES0. Otherwise, bits [M:0] might contain part of a fault address on a synchronous external abort.

On a management interrupt, PMBPTR_EL1 is frozen.

4.3.4 PMBSR_EL1, Profiling Buffer Status/syndrome Register

The PMBSR_EL1 characteristics are:

Purpose

Provides syndrome information to software when the buffer is disabled because the management interrupt has been raised.

Usage constraints

Direct reads and writes of PMBSR_EL1 are UNDEFINED at EL0.

Traps and enables

Subject to the exception prioritization rules:

- If MDCR_EL2.E2PB == `0b00` or `0b10`, then direct reads and writes of PMBSR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
- If MDCR_EL3.NSPB ≠ `0b01`, then direct reads and writes of PMBSR_EL1 at Secure EL1 generate a Trap exception to EL3.
- If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMBSR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

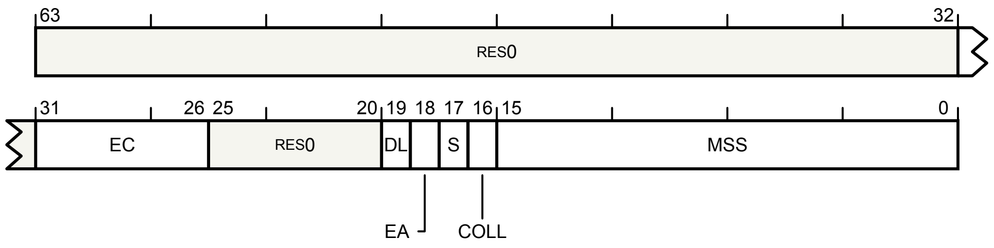
Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMBSR_EL1 are UNDEFINED otherwise.

Attributes

PMBSR_EL1 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_0_c9_c10_3`.

PMBSR_EL1

The PMBSR_EL1 bit assignments are:



Bits [63:32,25:20]

Reserved. This field is RES0.

EC, bits [31:26]

Exception class. Top-level description of the cause of the buffer management event. The possible values of this field are:

`0b100100`	Stage 1 Data Abort on write to Profiling Buffer.
`0b100101`	Stage 2 Data Abort on write to Profiling Buffer.
`0b000000`	Other buffer management event. All buffer management events other than those

described by other defined Exception class codes.

All other values are reserved. Reserved values might be defined in a future version of the architecture.

DL, bit [19]

Partial record lost. Following a buffer management event other than an asynchronous external abort, indicates whether the last record written to the Profiling Buffer is complete.

When the buffer management event was not because of an asynchronous external abort, the possible values of this bit are:

`0`	PMBPTR_EL1 points to the first byte after the last complete record written to the Profiling Buffer.
`1`	Part of a record was lost because of a buffer management event or synchronous external abort. PMBPTR_EL1 might not point to the first byte after the last complete record written to the buffer, and so restarting collection might result in a data record stream that software cannot parse. All records prior to the last record have been written to the buffer.

When the buffer management event was because of an asynchronous external abort, this bit is set to 1 and software must not assume that any valid data has been written to the Profiling Buffer.

EA, bit [18]

External abort. The possible values of this bit are:

`0`	An external abort has not been asserted.
`1`	An external abort has been asserted and detected by the Statistical Profiling Extension.

This bit is RES0 if the PE never sets this bit as the result of an external abort.

S, bit [17]

Service. The possible values of this bit are:

`0`	**PMBIRQ** is not asserted.
`1`	**PMBIRQ** is asserted. All profiling data has either been written to the buffer or discarded.

COLL, bit [16]

Collision detected. The possible values of this bit are:

`0`	No collision events detected.
`1`	At least one collision event was recorded.

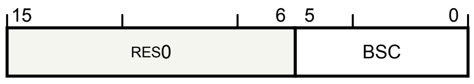
MSS, bits [15:0]

Management Event Specific Syndrome. Contains syndrome specific to the management event.

The syndrome contents for each management event are described in the following sections.

MSS encoding for other buffer management event (EC == `0b0000000`)

The MSS encoding for other buffer management event (EC == `0b0000000`) bit assignments are:



Bits [15:6]

Reserved. This field is RES0.

BSC, bits [5:0]

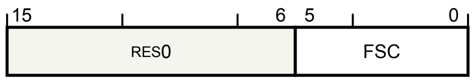
Buffer status code. The possible values of this field are:

`0b000000`	Buffer not filled.
`0b000001`	Buffer filled.

All other values are reserved. Reserved values might be defined in a future version of the architecture.

MSS encoding for data abort or stage 2 data abort on write to buffer (EC == `0b10010x`)

The MSS encoding for data abort or stage 2 data abort on write to buffer (EC == `0b10010x`) bit assignments are:



Bits [15:6]

Reserved. This field is RES0.

FSC, bits [5:0]

Fault status code. The possible values of this field are:

`0b0000xx`	Address Size fault. Bits [1:0] encode the level.
`0b0001xx`	Translation fault. Bits [1:0] encode the level.
`0b0010xx`	Access Flag fault. Bits [1:0] encode the level.
`0b0011xx`	Permission fault. Bits [1:0] encode the level.
`0b010000`	Synchronous external abort on write.
`0b0101xx`	Synchronous external abort on translation table walk or hardware update of translation table. Bits [1:0] encode the level.
`0b010001`	Asynchronous external abort on write.
`0b100001`	Alignment fault.
`0b110000`	TLB Conflict fault.
`0b110101`	Unsupported Access fault.

All other values are reserved. Reserved values might be defined in a future version of the architecture.

4.3.5 PMSCR_EL1, Statistical Profiling Control Register (EL1)

The PMSCR_EL1 characteristics are:

Purpose

Provides EL1 controls for Statistical Profiling.

Usage constraints

Direct reads and writes of PMSCR_EL1 are UNDEFINED at EL0.

If HCR_EL2.E2H==1, then permitted direct reads and writes using the register name PMSCR_EL1 at EL2 access PMSCR_EL2. Use PMSCR_EL12 to access PMSCR_EL1.

Traps and enables

Subject to the exception prioritization rules:

– If MDCR_EL2.TPMS == 1, then direct reads and writes of PMSCR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
– If MDCR_EL3.NSPB ≠ `0b01`, then direct reads and writes of PMSCR_EL1 at Secure EL1 generate a Trap exception to EL3.
– If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMSCR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

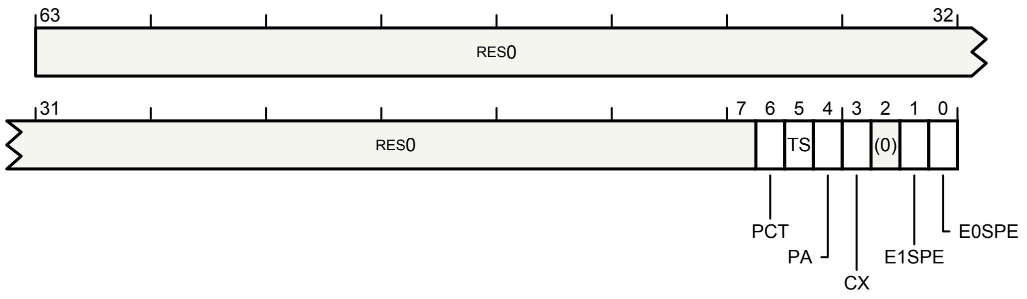
Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMSCR_EL1 are UNDEFINED otherwise.

Attributes

PMSCR_EL1 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_0_c9_c9_0`.

Field descriptions

The PMSCR_EL1 bit assignments are:



Bits [63:7,2]

Reserved. This field is RES0.

PCT, bit [6]

Physical Timestamp. If timestamp sampling is enabled, determines which counter is collected. The possible values of this bit are:

`0` Virtual counter, CNTVCT_EL0, is collected.

`1`	Physical counter, CNTPCT_EL0, is collected.

If MDCR_EL2.E2PB ≠ `0b00` and the PE is in Non-secure state, this bit is combined with PMSCR_EL2.PCT to determine which counter is collected.

This bit is RES1 if EL2 is not implemented. This bit is ignored by the PE when in Non-secure state and MDCR_EL2.E2PB == `0b00`.

See `CollectTimeStamp`.

TS, bit [5]

Timestamp Enable. The possible values of this bit are:

`0`	Timestamp sampling disabled.
`1`	Timestamp sampling enabled.

This bit is ignored by the PE when in Non-secure state and MDCR_EL2.E2PB == `0b00`.

See `CollectTimeStamp`.

PA, bit [4]

Physical Address Sample Enable. The possible values of this bit are:

`0`	Physical addresses are not collected.
`1`	Physical addresses are collected.

If MDCR_EL2.E2PB ≠ `0b00` and the PE is in Non-secure state, this bit is combined with PMSCR_EL2.PA to determine which address is collected.

This bit is ignored by the PE when in Non-secure state and MDCR_EL2.E2PB == `0b00`.

See `CollectPhysicalAddress`.

CX, bit [3]

CONTEXTIDR_EL1 Sample Enable. The possible values of this bit are:

`0`	CONTEXTIDR_EL1 is not collected.
`1`	CONTEXTIDR_EL1 is collected.

This bit is ignored by the PE when any of the following are true:

- At EL2.
- In Non-secure state and HCR_EL2.TGE == 1.

See `CollectContextIDR1`.

E1SPE, bit [1]

EL1 Statistical Profiling Enable. The possible values of this bit are:

`0`	Sampling disabled at EL1.
`1`	Sampling enabled at EL1.

This bit is ignored by the PE when in Non-secure state and HCR_EL2.TGE == 1.

See `StatisticalProfilingEnabled`.

E0SPE, bit [0]

EL0 Statistical Profiling Enable. The possible values of this bit are:

`0`	Sampling disabled at EL0.
`1`	Sampling enabled at EL0.

This bit is ignored by the PE when in Non-secure state and HCR_EL2.TGE == 1.

See `StatisticalProfilingEnabled`.

4.3.6 PMSCR_EL12, Statistical Profiling Control Register (EL1)

The PMSCR_EL12 characteristics are:

Purpose

Provides EL2 access to PMSCR_EL1.

Usage constraints

Direct reads and writes of PMSCR_EL12 are UNDEFINED when any of:

- At EL1 and EL0.
- HCR_EL2.E2H==0.
- EL3 is implemented and SCR_EL3.NS==0.

If HCR_EL2.E2H==1, then permitted direct reads and writes using the register name PMSCR_EL1 at EL2 access PMSCR_EL2. Use PMSCR_EL12 to access PMSCR_EL1.

Traps and enables

If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMSCR_EL12 at EL2 generate a Trap exception to EL3.

Configurations

Present only if all of the following apply:

- The Virtualization Host Extension is implemented.
- EL2 is implemented.
- The Statistical Profiling Extension is implemented.

Direct reads and writes of PMSCR_EL12 are UNDEFINED otherwise.

Attributes

PMSCR_EL12 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_5_c9_c9_0`.

4.3.7 PMSCR_EL2, Statistical Profiling Control Register (EL2)

The PMSCR_EL2 characteristics are:

Purpose

Provides EL2 controls for Statistical Profiling.

Usage constraints

Direct reads and writes of PMSCR_EL2 are UNDEFINED at EL1 and EL0.

If HCR_EL2.E2H==1, then permitted direct reads and writes using the register name PMSCR_EL1 at EL2 access PMSCR_EL2. Use PMSCR_EL12 to access PMSCR_EL1.

Traps and enables

If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMSCR_EL2 at EL2 generate a Trap exception to EL3.

Configurations

PMSCR_EL2 is RES0 at EL3 if EL2 is not implemented.

Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMSCR_EL2 are UNDEFINED otherwise.

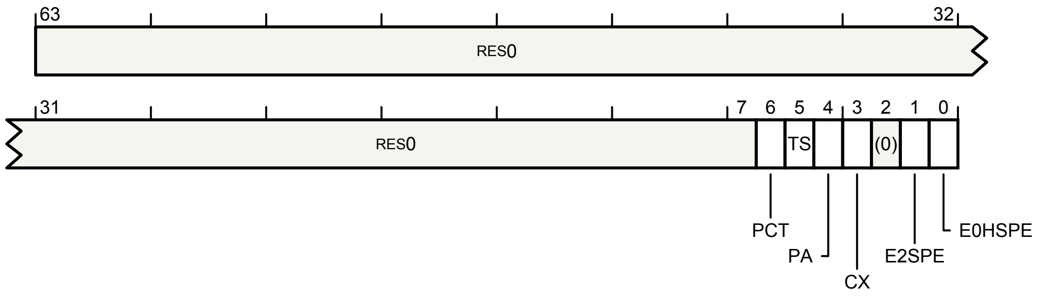
If EL2 is not implemented, then the PE behaves as if PA == 1 and PCT == 1, other than for a direct read of the register.

Attributes

PMSCR_EL2 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_4_c9_c9_0`.

Field descriptions

The PMSCR_EL2 bit assignments are:



Bits [63:7,2]

Reserved. This field is RES0.

PCT, bit [6]

Physical Timestamp. If timestamp sampling is enabled, determines which counter is collected. The possible values of this bit are:

`0`	Virtual counter, CNTVCT_EL0, is collected.
`1`	Physical counter, CNTPCT_EL0, is collected.

If MDCR_EL2.E2PB ≠ `0b00`, this bit is combined with PMSCR_EL1.PCT to determine which counter is collected.

This bit is ignored by the PE when in Secure state. If EL2 is not implemented, the PE behaves as if this bit is set to 1, other than for a direct read of the register.

See `CollectTimeStamp`.

TS, bit [5]

Timestamp Enable. The possible values of this bit are:

`0`	Timestamp sampling disabled.
`1`	Timestamp sampling enabled.

This bit is ignored by the PE when any of the following are true:

- In Non-secure state and MDCR_EL2.E2PB ≠ `0b00`.
- In Secure state.

See `CollectTimeStamp`.

PA, bit [4]

Physical Address Sample Enable. The possible values of this bit are:

`0`	Physical addresses are not collected.
`1`	Physical addresses are collected.

If MDCR_EL2.E2PB ≠ `0b00`, this bit is combined with PMSCR_EL1.PA to determine which address is collected.

This bit is ignored by the PE when in Secure state. If EL2 is not implemented, the PE behaves as if this bit is set to 1, other than for a direct read of the register.

See `CollectPhysicalAddress`.

CX, bit [3]

CONTEXTIDR_EL2 Sample Enable. The possible values of this bit are:

`0`	CONTEXTIDR_EL2 is not collected.
`1`	CONTEXTIDR_EL2 is collected.

This bit is ignored by the PE when in Secure state.

See `CollectContextIDR2`.

E2SPE, bit [1]

EL2 Statistical Profiling Enable. The possible values of this bit are:

`0`	Sampling disabled at EL2.
`1`	Sampling enabled at EL2.

This bit is RES0 if MDCR_EL2.E2PB ≠ `0b00`. This bit is ignored by the PE when in Secure state.

See `StatisticalProfilingEnabled`.

E0HSPE, bit [0]

EL0 Statistical Profiling Enable. The possible values of this bit are:

`0` Sampling disabled at EL0.

`1` Sampling enabled at EL0.

This bit is RES0 if MDCR_EL2.E2PB ≠ `0b00`.

This bit is ignored by the PE when any of the following are true:

- In Non-secure state and HCR_EL2.TGE == 0.
- In Secure state.

See `StatisticalProfilingEnabled`.

4.3.8 PMSEVFR_EL1, Sampling Event Filter Register

The PMSEVFR_EL1 characteristics are:

Purpose

Controls sample filtering by events. The overall filter is the logical AND of these filters. For example, if E[3] and E[5] are both set to 1, only samples that have both event 3 (Level 1 unified or data cache refill) and event 5 set (TLB walk) are recorded.

Usage constraints

Direct reads and writes of PMSEVFR_EL1 are UNDEFINED at EL0.

Traps and enables

Subject to the exception prioritization rules:

- If MDCR_EL2.TPMS == 1, then direct reads and writes of PMSEVFR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
- If MDCR_EL3.NSPB ≠ `0b01`, then direct reads and writes of PMSEVFR_EL1 at Secure EL1 generate a Trap exception to EL3.
- If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMSEVFR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

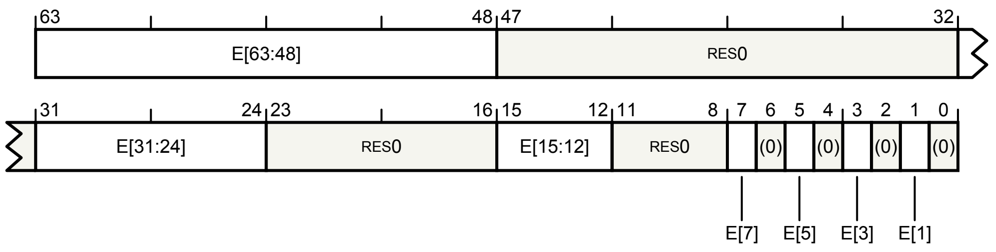
Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMSEVFR_EL1 are UNDEFINED otherwise.

Attributes

PMSEVFR_EL1 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_0_c9_c9_5`.

Field descriptions

The PMSEVFR_EL1 bit assignments are:



E[63:48,31:24,15:12], bits [63:48,31:24,15:12]

E[<*n*>] is the event filter for event <*n*>. If event <*n*> is not implemented, or filtering on event <*n*> is not supported, the corresponding bit is RES0. The possible values of this field are:

`0`	Event <*n*> is ignored.
`1`	Record samples that have event <*n*> == 1.

An IMPLEMENTATION DEFINED event might be recorded as a multi-bit field. In this case, if the corresponding bits of PMSEVFR_EL1 define an IMPLEMENTATION DEFINED filter for the event.

This field is ignored by the PE when PMSFCR_EL1.FE == 0.

Bits [47:32,23:16,11:8,6,4,2,0]

Reserved. This field is RES0.

E[7], bit [7]

Mispredicted. The possible values of this bit are:

`0` Mispredicted event is ignored.

`1` Record samples that have event 7 (Mispredicted) == 1.

This bit is ignored by the PE when PMSFCR_EL1.FE == 0.

E[5], bit [5]

TLB walk. The possible values of this bit are:

`0` TLB walk event is ignored.

`1` Record samples that have event 5 (TLB walk) == 1.

This bit is ignored by the PE when PMSFCR_EL1.FE == 0.

E[3], bit [3]

Level 1 data or unified cache refill. The possible values of this bit are:

`0` Level 1 data or unified cache refill event is ignored.

`1` Record samples that have event 3 (Level 1 data or unified cache refill) == 1.

This bit is ignored by the PE when PMSFCR_EL1.FE == 0.

E[1], bit [1]

Architecturally retired. The possible values of this bit are:

`0` Architecturally retired event is ignored.

`1` Record samples that have event 1 (Architecturally retired) == 1.

This bit is RES1 if the PE does not support sampling of speculative instructions. This bit is ignored by the PE when PMSFCR_EL1.FE == 0.

4.3.9 PMSFCR_EL1, Sampling Filter Control Register

The PMSFCR_EL1 characteristics are:

Purpose

Controls sample filtering. The filter is the logical AND of the FL, FT and FE bits. For example, if FE == 1 and FT == 1 only samples including the selected operation types and the selected events will be recorded.

Usage constraints

Direct reads and writes of PMSFCR_EL1 are UNDEFINED at EL0.

Traps and enables

Subject to the exception prioritization rules:

- If MDCR_EL2.TPMS == 1, then direct reads and writes of PMSFCR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
- If MDCR_EL3.NSPB ≠ `0b01`, then direct reads and writes of PMSFCR_EL1 at Secure EL1 generate a Trap exception to EL3.
- If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMSFCR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

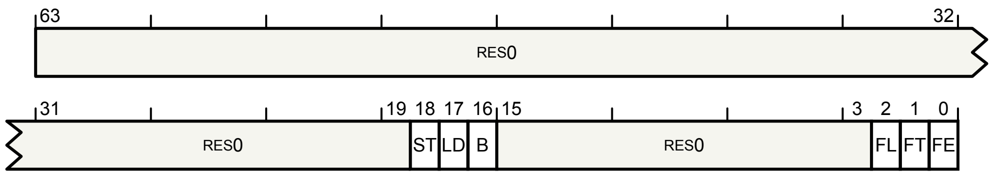
Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMSFCR_EL1 are UNDEFINED otherwise.

Attributes

PMSFCR_EL1 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_0_c9_c9_4`.

Field descriptions

The PMSFCR_EL1 bit assignments are:



Bits [63:19,15:3]

Reserved. This field is RES0.

ST, bit [18]

Store filter enable. The possible values of this bit are:

`0`	Do not record store operations.
`1`	Record all store operations, including vector stores and all atomic operations.

This bit is ignored by the PE when PMSFCR_EL1.FT == 0.

LD, bit [17]

Load filter enable. The possible values of this bit are:

`0` Do not record load operations.

`1` Record all load operations, including vector loads and atomic operations that return data.

This bit is ignored by the PE when PMSFCR_EL1.FT == 0.

B, bit [16]

Branch filter enable. The possible values of this bit are:

`0` Do not record branch and exception return operations.

`1` Record all branch and exception return operations.

This bit is ignored by the PE when PMSFCR_EL1.FT == 0.

FL, bit [2]

Filter by latency. The possible values of this bit are:

`0` Latency filtering disabled.

`1` Latency filtering enabled. Samples with a total latency less than PMSLATFR_EL1.MINLAT will not be recorded.

If this bit is set to 1 and PMSLATFR_EL1.MINLAT is set to zero, it is CONSTRAINED UNPREDICTABLE whether no samples are recorded or the PE behaves as if PMSFCR_EL1.FL is set to 0.

See `CollectRecord`.

FT, bit [1]

Filter by operation type. The filter is the logical OR of the ST, LD and B bits. For example, if LD and ST are both set, both load and store operations are recorded. The possible values of this bit are:

`0` Type filtering disabled.

`1` Type filtering enabled. Samples not one of the selected operation types will not be recorded.

If this bit is set to 1 and the PMSFCR_EL1.{ST, LD, B} bits are all set to zero, it is CONSTRAINED UNPREDICTABLE whether no samples are recorded or the PE behaves as if PMSFCR_EL1.FT is set to 0.

See `CollectRecord`.

FE, bit [0]

Filter by event. The possible values of this bit are:

`0` Event filtering disabled.

`1` Event filtering enabled. Samples not including the events selected by PMSEVFR_EL1 will not be recorded.

If this bit is set to 1 and PMSEVFR_EL1 is set to zero, it is CONSTRAINED UNPREDICTABLE whether no samples are recorded or the PE behaves as if PMSFCR_EL1.FE is set to 0.

See `CollectRecord`.

4.3.10 PMSICR_EL1, Sampling Interval Counter Register

The PMSICR_EL1 characteristics are:

Purpose

Software must write zero to PMSICR_EL1 before enabling sample profiling for a sampling session. Software must then treat PMSICR_EL1 as an opaque, 64-bit, read/write register used for context switches only.

Usage constraints

Direct reads and writes of PMSICR_EL1 are UNDEFINED at EL0.

Traps and enables

Subject to the exception prioritization rules:

- If MDCR_EL2.TPMS == 1, then direct reads and writes of PMSICR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
- If MDCR_EL3.NSPB ≠ `0b01`, then direct reads and writes of PMSICR_EL1 at Secure EL1 generate a Trap exception to EL3.
- If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMSICR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMSICR_EL1 are UNDEFINED otherwise.

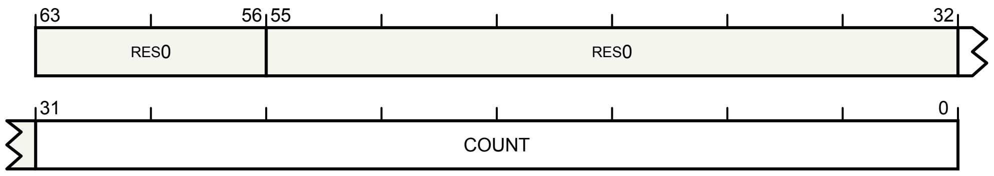
Attributes

PMSICR_EL1 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_0_c9_c9_2`.

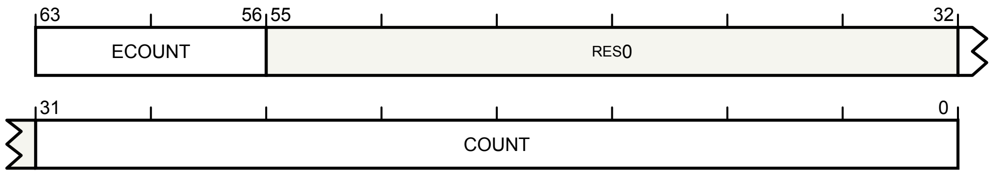
The value of PMSICR_EL1 does not change whilst profiling is disabled.

Field descriptions

When PMSIDR_EL1.ERnd == 0, the PMSICR_EL1 bit assignments are:



When PMSIDR_EL1.ERnd == 1, the PMSICR_EL1 bit assignments are:



ECOUNT, bits [63:56], when PMSIDR_EL1.ERnd == 1

Secondary sample interval counter. Provides the secondary counter used after the primary counter reaches zero. Whilst the secondary counter is nonzero and profiling is enabled, the secondary counter decrements by 1 for each member of the sample population. The primary counter also continues to decrement since it is also nonzero. When the secondary counter reaches zero, a member of the sampling population is selected for sampling.

Bits [63:56], when PMSIDR_EL1.ERnd == 0

Reserved. This field is RES0.

Bits [55:32]

Reserved. This field is RES0.

COUNT, bits [31:0]

Primary sample interval counter.

Provides the primary counter used for sampling.

The primary counter is reloaded when the value of this register is zero and the PE moves from a state or Exception level where profiling is disabled to a state or Exception level where profiling is enabled.

Whilst the primary counter is nonzero and sampling is enabled, the primary counter decrements by 1 for each member of the sample population.

When the counter reaches zero, the behavior depends on the values of PMSIDR_EL1.ERnd and PMSIRR_EL1.RND.

- If PMSIRR_EL1.RND == 0 or PMSIDR_EL1.ERnd == 0:
  - A member of the sampling population is selected for sampling.
  - The primary counter is reloaded.
- If PMSIRR_EL1.RND == 1 and PMSIDR_EL1.ERnd == 1:
  - The secondary counter is set to a random or pseudorandom value in the range `0x00` to `0xFF`.
  - The primary counter is reloaded.

See Controlling when a sample is taken.

4.3.11 PMSIDR_EL1, Sampling Profiling ID Register

The PMSIDR_EL1 characteristics are:

Purpose

Describes the Statistical Profiling implementation to software.

Usage constraints

Direct reads of PMSIDR_EL1 are UNDEFINED at EL0.

Traps and enables

Subject to the exception prioritization rules:

- If MDCR_EL2.TPMS == 1, then direct reads of PMSIDR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
- If MDCR_EL3.NSPB ≠ `0b01`, then direct reads of PMSIDR_EL1 at Secure EL1 generate a Trap exception to EL3.
- If MDCR_EL3.NSPB ≠ `0b11`, then direct reads of PMSIDR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

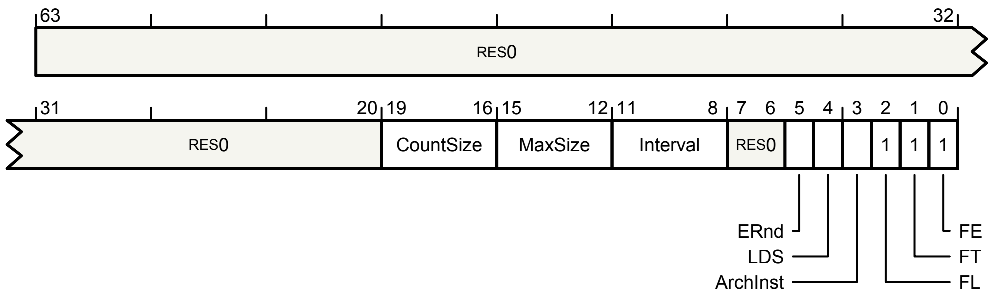
Present only if the Statistical Profiling Extension is implemented. Direct reads of PMSIDR_EL1 are UNDEFINED otherwise.

Attributes

PMSIDR_EL1 is a 64-bit read-only AArch64 System register accessed using `MRS` of `S3_0_c9_c9_7`.

Field descriptions

The PMSIDR_EL1 bit assignments are:



Bits [63:20,7:6]

Reserved. This field is RES0.

CountSize, bits [19:16]

Defines the size of the counters. The defined values of this field are:

`0b0010` 12-bit saturating counters.

All other values are reserved. Reserved values might be defined in a future version of the architecture.

MaxSize, bits [15:12]

Defines the largest size for a single record, rounded up to a power-of-two. If this is the same as the minimum alignment (PMBIDR_EL1.Align), then each record is exactly this size. The defined values of this field are:

`0b0100`	16 bytes.
`0b0101`	32 bytes.
`0b0110`	64 bytes.
`0b0111`	128 bytes.
...	.
`0b1011`	2KB.

All other values are reserved. Reserved values might be defined in a future version of the architecture.

Interval, bits [11:8]

Recommended minimum sampling interval. This provides guidance from the implementer to the smallest minimum sampling interval, N. The defined values of this field are:

`0b0000`	256.
`0b0010`	512.
`0b0011`	768.
`0b0100`	1,024.
`0b0101`	1,536.
`0b0110`	2,048.
`0b0111`	3,072.
`0b1000`	4,096.

All other values are reserved. Reserved values might be defined in a future version of the architecture.

ERnd, bit [5]

Defines how the random number generator is used in determining the interval between samples, when enabled by PMSIRR_EL1.RND. The defined values of this bit are:

`0`	The random number is added at the start of the interval, and the sample is taken and a new interval started when the combined interval expires.
`1`	The random number is added and the new interval started after the interval programmed in PMSIRR_EL1.INTERVAL expires, and the sample is taken when the random interval expires.

LDS, bit [4]

Data source indicator for sampled load instructions. The defined values of this bit are:

`0`	Loaded data source not implemented.

`1` Loaded data source implemented.

ArchInst, bit [3]

Architectural instruction profiling. The defined values of this bit are:

`0` Micro-op sampling implemented.

`1` Architecture instruction sampling implemented.

FL, bit [2]

Filtering by latency. This bit reads as one.

FT, bit [1]

Filtering by operation type. This bit reads as one.

FE, bit [0]

Filtering by events. This bit reads as one.

4.3.12 PMSIRR_EL1, Sampling Interval Reload Register

The PMSIRR_EL1 characteristics are:

Purpose

Defines the interval between samples.

Usage constraints

Direct reads and writes of PMSIRR_EL1 are UNDEFINED at EL0.

Traps and enables

Subject to the exception prioritization rules:

- If MDCR_EL2.TPMS == 1, then direct reads and writes of PMSIRR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
- If MDCR_EL3.NSPB ≠ `0b01`, then direct reads and writes of PMSIRR_EL1 at Secure EL1 generate a Trap exception to EL3.
- If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMSIRR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

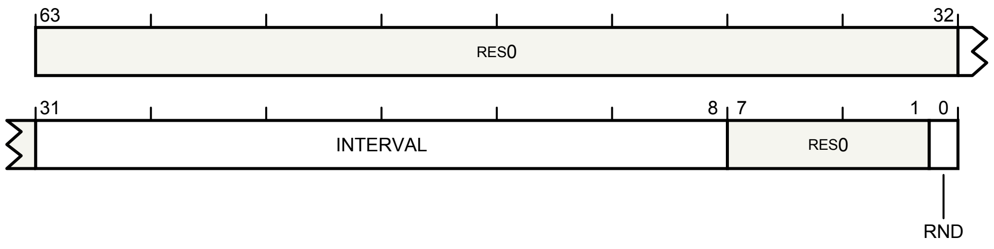
Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMSIRR_EL1 are UNDEFINED otherwise.

Attributes

PMSIRR_EL1 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_0_c9_c9_3`.

Field descriptions

The PMSIRR_EL1 bit assignments are:



Bits [63:32,7:1]

Reserved. This field is RES0.

INTERVAL, bits [31:8]

Bits [31:8] of the PMSICR_EL1 interval counter reload value. Software must set this to a non-zero value. If software sets this to zero, an UNKNOWN sampling interval is used. Software should set this to a value greater than the minimum indicated by PMSIDR_EL1.Interval.

RND, bit [0]

Controls randomization of the sampling interval. The possible values of this bit are:

`0`	Disable randomization of sampling interval.

`1` Add (pseudo-)random jitter to sampling interval.

The random number generator is not architected.

4.3.13 PMSLATFR_EL1, Sampling Latency Filter Register

The PMSLATFR_EL1 characteristics are:

Purpose

Controls sample filtering by latency.

Usage constraints

Direct reads and writes of PMSLATFR_EL1 are UNDEFINED at EL0.

Traps and enables

Subject to the exception prioritization rules:

- If MDCR_EL2.TPMS == 1, then direct reads and writes of PMSLATFR_EL1 at Non-secure EL1 generate a Trap exception to EL2.
- If MDCR_EL3.NSPB ≠ `0b01`, then direct reads and writes of PMSLATFR_EL1 at Secure EL1 generate a Trap exception to EL3.
- If MDCR_EL3.NSPB ≠ `0b11`, then direct reads and writes of PMSLATFR_EL1 at EL2 and Non-secure EL1 generate a Trap exception to EL3.

Configurations

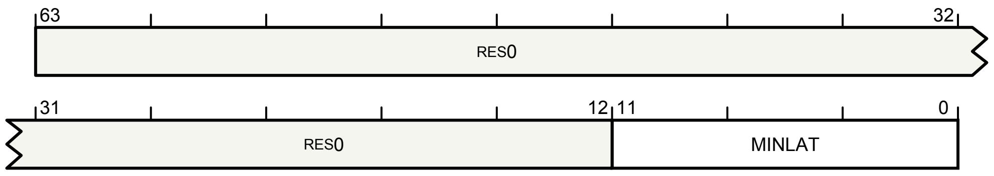
Present only if the Statistical Profiling Extension is implemented. Direct reads and writes of PMSLATFR_EL1 are UNDEFINED otherwise.

Attributes

PMSLATFR_EL1 is a 64-bit read/write AArch64 System register accessed using `MRS` and `MSR` of `S3_0_c9_c9_6`.

Field descriptions

The PMSLATFR_EL1 bit assignments are:



Bits [63:12]

Reserved. This field is RES0.

MINLAT, bits [11:0]

Minimum latency. When PMSFCR_EL1.FL == 1, defines the minimum total latency for filtered operations. Samples with a total latency less than MINLAT will not be recorded.

This field is ignored by the PE when PMSFCR_EL1.FL == 0.

See `CollectRecord`.

5 Profile Format

5.1 Overview

The profile data is self-describing and extensible. This format allows for the parsing of profile data even when that profile data contains extended information.

5.1.1 Packets

The profile data is written as sample records, each sample record consisting of a sequence of *packets*, and each packet consisting of:

- One or two header bytes.
- Zero, 1, 2, 4 or 8 payload bytes. The number of payload bytes is encoded in the header byte.

Figure 1 shows the basic packet format with a single byte (8-bit) header:

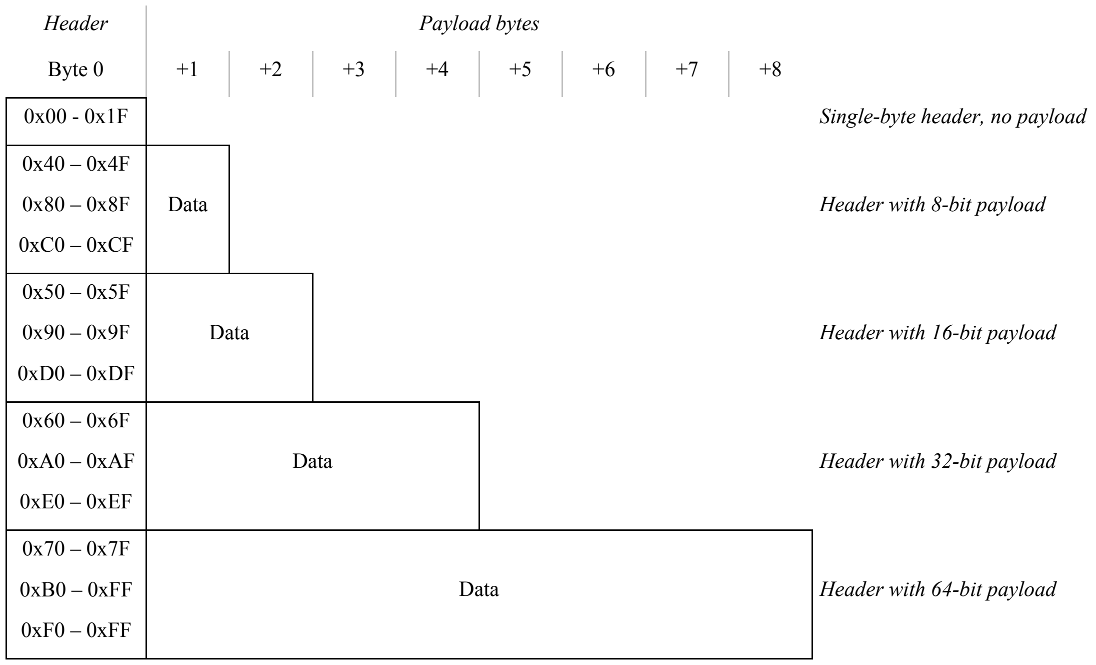


Figure 1: Basic packet types

Figure 2 shows the extended packet formats with 16-bit headers, that is, including prefix bytes. In these packets, the prefix byte extends the header in some way.

Header bytes		*Payload bytes*								
Byte 0	+1	+2	+3	+4	+5	+6	+7	+8	+9	
0x20 – 0x3F	0x00 - 0x1F									*Extended byte header, no payload*
0x20 – 0x3F	0x20 – 0x3F									Reserved
0x20 – 0x3F	0x40 – 0x4F 0x80 – 0x8F 0xC0 – 0xCF	Data								*Extended header with 8-bit payload*
0x20 – 0x3F	0x50 – 0x5F 0x90 – 0x9F 0xD0 – 0xDF	Data								*Extended header with 16-bit payload*
0x20 – 0x3F	0x60 – 0x6F 0xA0 – 0xAF 0xE0 – 0xEF	Data								*Extended header with 32-bit payload*
0x20 – 0x3F	0x70 – 0x7F 0xB0 – 0xFF 0xF0 – 0xFF	Data								*Extended header with 64-bit payload*

Figure 2: Extended packet types

5.1.2 Records

A *sample record* consists of multiple *packets*. A sample record comprises, in ascending address order:

- A sequence of headers, each followed by their payload bytes.
- Either:
  - An end packet header.
  - A timestamp packet.

First byte	**1**	**2**	**3**	**4**	**Last byte**
Header (16-bit data)	*Data* *LSB*	*MSB*	**Header** (8-bit data)	*Data*	**0x01** End packet

Figure 3: Example record with end packet

First byte	**1**	**2**	**3**	**4**	**5**	**6**	**…**	**12**	**Last byte**
Header (16-bit data)	*Data* *LSB*	*MSB*	**Header** (8-bit data)	*Data*	**0x71** Timestamp packet	TS[7:0]	…	TS[55:48]	TS[63:56]

Figure 4: Example record with timestamp packet

5.1.3 Byte order

Header bytes and data bytes are written in ascending address order. Within a data value, values are written in little-endian byte order.

5.1.4 Protocol framing packets and forwards compatibility

The Padding header, Alignment command, Timestamp packet, and End packet are protocol framing packets that frame the records created by the Statistical Profiling Extension.

Only Padding headers and Alignment commands are permitted between sample records.

Note

PMBIDR_EL1.Align defines a minimum alignment for sample records. However, implementations must nevertheless create a valid protocol stream that can be parsed without knowledge of the minimum alignment.

The packet types are described in the following sections. Software must ignore unknown packets, using the size field that is encoded in the header. This includes packets containing reserved values in fields.

5.2 Top-level encodings

5.2.1 8-bit headers

[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]	Description
0	0	0	0	0	0	0	0	Padding
0	0	0	0	0	0	0	1	End packet
0	1	1	1	0	0	0	1	Timestamp packet
0	1	x	x	0	0	1	0	Events packet
0	1	x	x	0	0	1	1	Data Source packet
0	1	1	0	0	1	x	x	Context packet
0	1	0	0	1	0	x	x	Operation Type packet
1	0	1	1	0	x	x	x	Address packet (Short format)
1	0	0	1	1	x	x	x	Counter packet (Short format)

Figure 5: 8-bit header encodings

5.2.2 16-bit headers

Byte 0								Byte 1								Description
[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]	[7]	[6]	[5]	[4]	[3]	[2]	[1]	[0]	
0	0	1	0	x	x	x	x	0	0	0	0	0	0	0	0	Alignment command
0	0	1	0	0	0	x	x	1	0	1	1	0	x	x	x	Address packet (Extended format)
0	0	1	0	0	0	x	x	1	0	0	1	1	x	x	x	Counter packet (Extended format)

Figure 6: 16-bit header encodings

5.3 Packets

5.3.1 Address packet

The Address packet characteristics are:

Purpose

Provides an address value for the record. Addresses are always 64 bits.

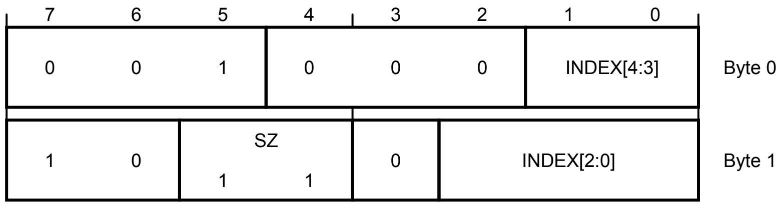
Attributes

Multi-part packet comprising:

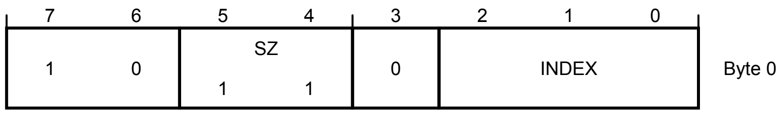
- 8 or 16-bit header.
- 64-bit payload.

Address packet header

When Extended format, the Address packet header bit assignments are:



When Short format, the Address packet header bit assignments are:



Byte 1 bits [7:6], when Extended format

Byte 0 bits [7:6], when Short format

This field reads as `0b10`.

SZ, byte 1 bits [5:4], when Extended format

SZ, byte 0 bits [5:4], when Short format

Payload size. The defined values of this field are:

`0b11`	Doubleword.

This field reads as `0b11`.

Byte 1 bit [3], when Extended format

Byte 0 bit [3], when Short format

This bit reads as `0b0`.

Byte 0 bits [7:5], when Extended format

This field reads as `0b001`.

Byte 0 bits [4:2], when Extended format

This field reads-as-zero.

INDEX, byte 0 bits [1:0], byte 1 bits [2:0], when Extended format

INDEX, byte 0 bits [2:0], when Short format

The defined values of this field are:

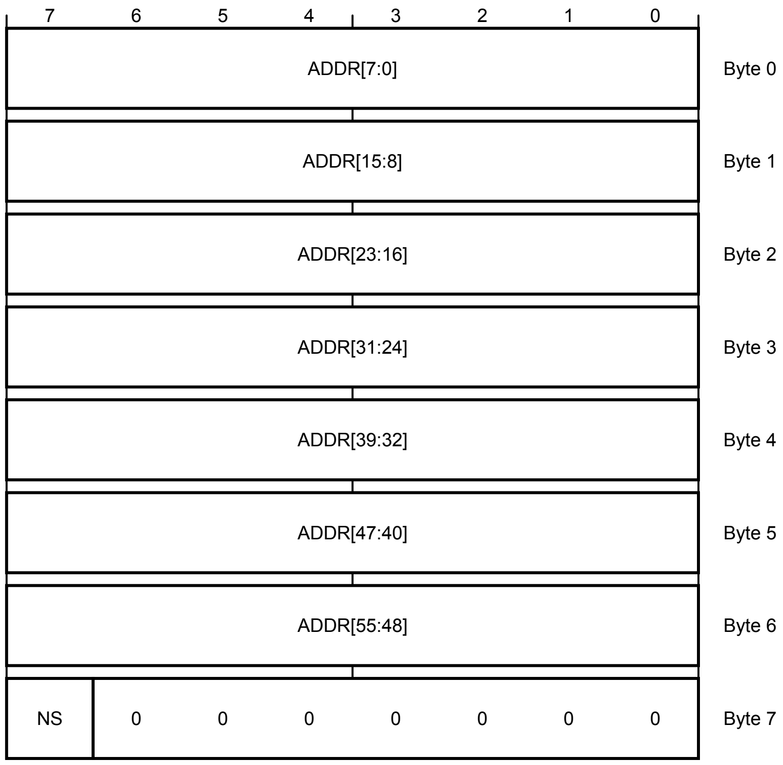
`0b00000`	Issued instruction virtual address (PC). Included for all operations.
`0b00001`	Branch target address: • It is IMPLEMENTATION DEFINED and might be UNPREDICTABLE whether this address is included for an Exception Return to an Exception level where profiling is disabled. • Included for all other branch and exception return instructions.
`0b00010`	Data access virtual address. Included for all load, store and atomic operations.
`0b00011`	Data access physical address: • It is IMPLEMENTATION DEFINED and might be UNPREDICTABLE whether this address included for accesses that generate Permission or Access Flag faults. • Not included for all other accesses that generate an abort, or if disabled by `CollectPhysicalAddress`. • Included for all other load, store and atomic operations.
`0b0011x`	IMPLEMENTATION DEFINED address.
`0b1xxxx`	IMPLEMENTATION DEFINED address.

All other values are reserved.

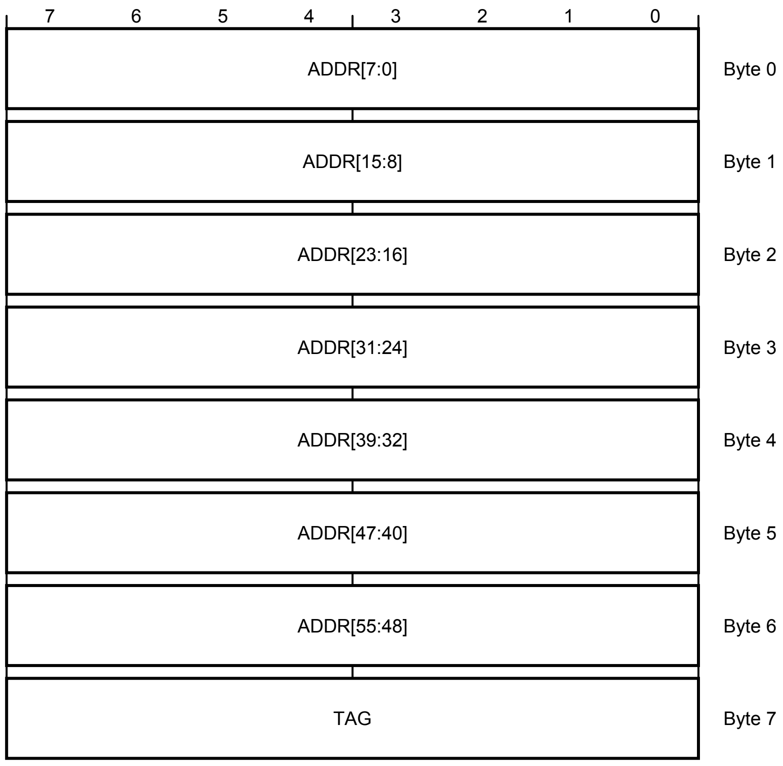
In the Short format header, bits [4:3] are zero.

Address packet payload

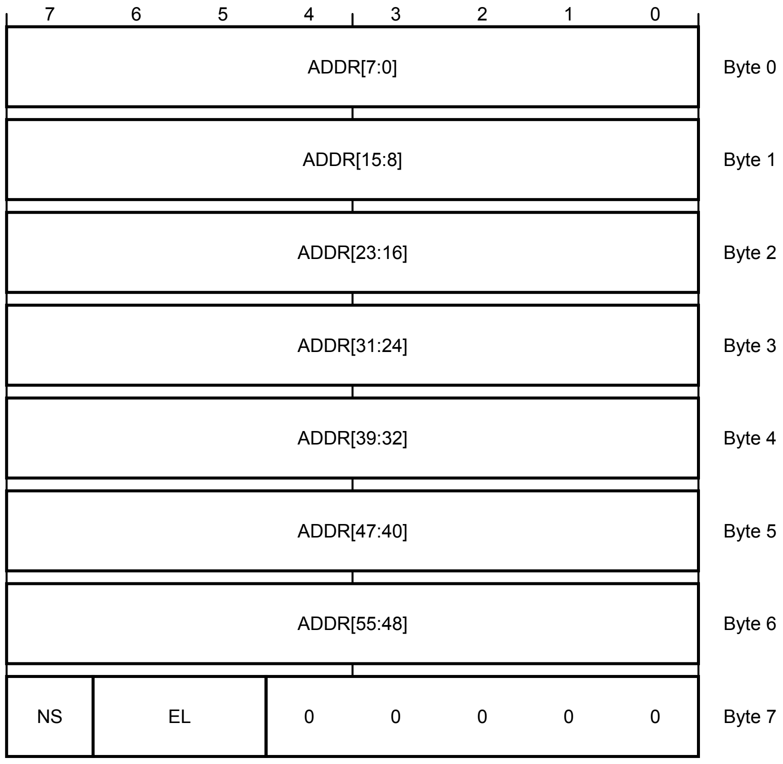
When Data access physical address, the Address packet payload bit assignments are:



When Data access virtual address, the Address packet payload bit assignments are:



When Instruction virtual address, the Address packet payload bit assignments are:



TAG, byte <7>, when Data access virtual address

Top-byte tag.

If the applicable TBI bit is set to 1, a data access virtual address includes the top-byte tag. Otherwise this field reads-as-zero.

NS, byte 7 bit [7], when Instruction virtual address

Non-secure state. The Security state associated with the address. For an issued instruction virtual address (PC) this is the Security state the instruction was executed in. For a branch target address, this is the Security state at the target of the branch. The defined values of this bit are:

`0`	Secure state.
`1`	Non-secure state.

Note

For an Exception Return, the Security state at the target of the branch might be different to the Security state the instruction was executed in.

NS, byte 7 bit [7], when Data access physical address

Physical address space identifier. The Security attribute for the physical address. The defined values of this bit are:

`0`	Secure physical address space.
`1`	Non-secure physical address space.

EL, byte 7 bits [6:5], when Instruction virtual address

Exception level. The Exception level associated with the address. For an issued instruction virtual address (PC) this is the Exception level the instruction was executed in. For a branch target address, this is the Exception level at the target of the branch. The defined values of this field are:

`0b00`	EL0.
`0b01`	EL1.
`0b10`	EL2.
`0b11`	EL3.

Note

For an Exception Return, the Exception level at the target of the branch might be different to the Exception level the instruction was executed in.

Byte 7 bits [6:0], when Data access physical address

This field reads as `0b0000000`.

Byte 7 bits [4:0], when Instruction virtual address

This field reads as `0b00000`.

ADDR, bytes <6:0>

Address. Bits [55:0] of the address.

5.3.2 Alignment command

The Alignment command characteristics are:

Purpose

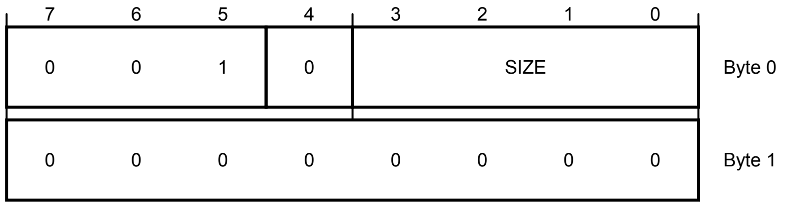
Allows the PE to create alignment to a naturally aligned address boundary in the buffer for the next protocol byte. Alignment bytes must be ignored.

Attributes

16-bit packet.

Field descriptions

The Alignment command bit assignments are:



Byte <1>

This field reads as `0b00000000`.

Byte 0 bits [7:5]

This field reads as `0b001`.

Byte 0 bit [4]

This bit reads-as-zero.

SIZE, byte 0 bits [3:0]

Defines the alignment of the next protocol byte. The data between the Alignment command and the next header byte is UNKNOWN. The defined values of this field are:

`0b0001`	4 byte alignment.
`0b0010`	8 byte alignment.
`0b0011`	16 byte alignment.
...	...
`0b1111`	64 Kbyte alignment.

All other values are reserved.

—— Note ——

The address of the next protocol byte is aligned to a multiple of the specified size. There will always be fewer than this number of bytes between the Alignment command and the next protocol byte.

5.3.3 Context packet

The Context packet characteristics are:

Purpose

Provides context information for the record.

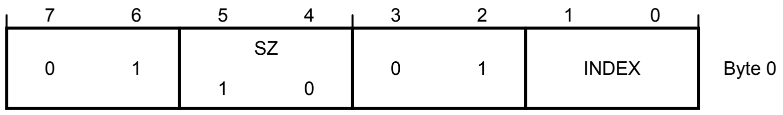
Attributes

Multi-part packet comprising:

- 8-bit header.
- 32-bit payload.

Context packet header

The Context packet header bit assignments are:



Byte 0 bits [7:6]

This field reads as `0b01`.

SZ, byte 0 bits [5:4]

Payload size. The defined values of this field are:

`0b10`	Word.

This field reads as `0b10`.

Byte 0 bits [3:2]

This field reads as `0b01`.

INDEX, byte 0 bits [1:0]

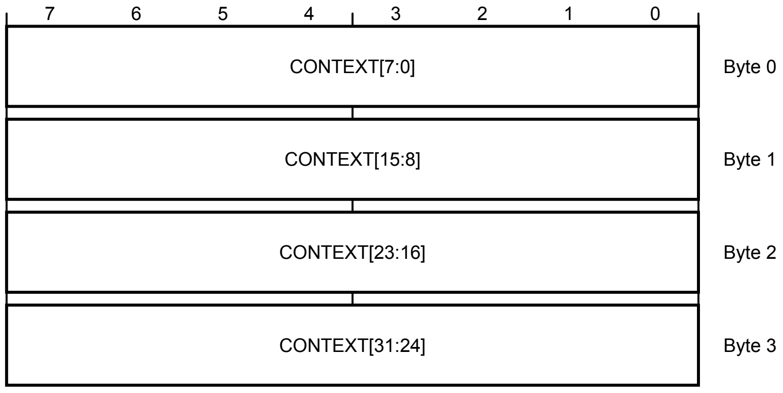
Identifies the context value. The defined values of this field are:

`0b00`	CONTEXTIDR_EL1. Included for all operations if enabled by `CollectContextIDR1`.
`0b01`	CONTEXTIDR_EL2. Included for all operations in Non-secure state if enabled by `CollectContextIDR2`.

All other values are reserved.

Context packet payload

The Context packet payload bit assignments are:



CONTEXT, bytes <3:0>

The context value.

5.3.4 Counter packet

The Counter packet characteristics are:

Purpose

Count of cycles the operation spent performing all or part of its behavior. The counter value occupies the least significant bits of the payload. The remaining bits are set to zero.

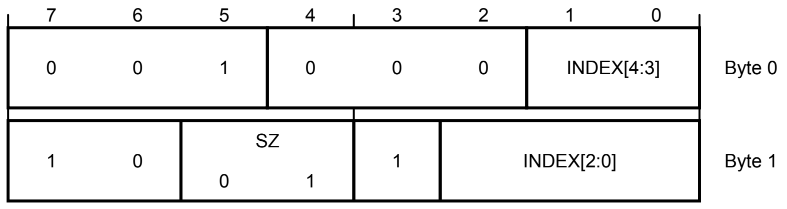
Attributes

Multi-part packet comprising:

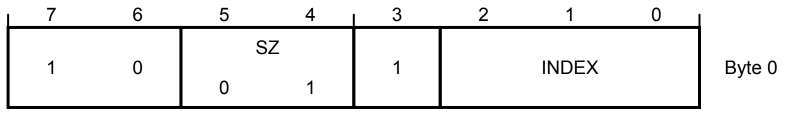
- 8 or 16-bit header.
- 16-bit payload.

Counter packet header

When Extended format, the Counter packet header bit assignments are:



When Short format, the Counter packet header bit assignments are:



Byte 1 bits [7:6], when Extended format

Byte 0 bits [7:6], when Short format

This field reads as `0b10`.

SZ, byte 1 bits [5:4], when Extended format

SZ, byte 0 bits [5:4], when Short format

Payload size. The defined values of this field are:

`0b01`	Halfword.

This field reads as `0b01`.

Byte 1 bit [3], when Extended format

Byte 0 bit [3], when Short format

This bit reads as `0b1`.

Byte 0 bits [7:5], when Extended format

This field reads as `0b001`.

Byte 0 bits [4:2], when Extended format

This field reads-as-zero.

INDEX, byte 0 bits [1:0], byte 1 bits [2:0], when Extended format

INDEX, byte 0 bits [2:0], when Short format

The defined values of this field are:

`0b00000`	Total latency. Cycle count from the operation being dispatched for issue to the operation being complete. Included for all operations.
`0b00001`	Issue latency. Cycle count from the operation being dispatched for issue to the operation being issued for execution. This counts any delay in waiting the operation being ready to issue. Included for all operations.
`0b00010`	Translation latency. Cycle count from a virtual address being passed to the MMU for translation to the result of the translation being available. Included for all load, store and atomic operations.
`0b0011x`	IMPLEMENTATION DEFINED counter value.
`0b1xxxx`	IMPLEMENTATION DEFINED counter value.

All other values are reserved.

In the Short format header, bits [4:3] are zero.

Dispatched for issue means:

- The operation has been decoded.
- The operation might not be ready to start execution because it is waiting for input values. The operation might be put into a queue.

Issued for execution means the operation is ready to start executing:

- For example, for a memory operation, this should be indicative of the cycle count from memory operation being dispatched for issue to access being initiated (virtual address).

Complete means:

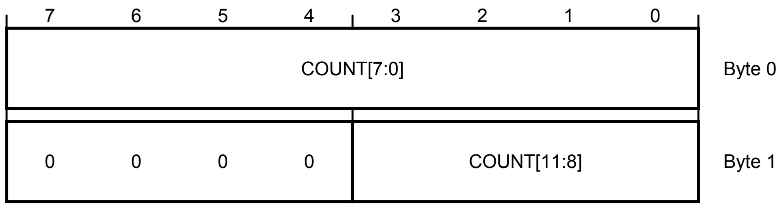
- The operation has completed execution and is no longer capable of stalling any instruction that consumes its output.
- It is IMPLEMENTATION DEFINED whether the operation has committed its results to the architectural state of the PE.
- For example:
  - For an arithmetic, FP or SIMD operation with variable timing, such as divide the results of the operation are available.
  - For load and atomic operations that return data, all data have been returned from memory.
  - For store and atomic operations that do not return data, it is not required that the store has reached its end point for it to be complete.
  - For branch operations, the branch has been resolved as taken or not taken.
  - For barrier operations, the barrier has completed.

For WFE and WFI operations, it is IMPLEMENTATION DEFINED whether:

- The instruction is complete before the PE enters a low-power state or when the PE wakes from the low-power state.
- Counters count in the low power state.
- Sampling an operation is itself a wake-up event.

Counter packet payload

The Counter packet payload bit assignments are:



Byte 1 bits [7:4]

This field reads-as-zero.

COUNT, byte 1 bits [3:0], byte <0>, when a 12-bit counter is implemented

The counter value occupies the least significant bits of the payload. The remaining bits are set to zero. The counters are:

- Unsigned numbers.
- 12 bits.
- Saturating.

The value `0xFFF` indicates the count has saturated.

5.3.5 Data Source packet

The Data Source packet characteristics are:

Purpose

If the implementation includes support for indicating the loaded data source, the Data Source packet indicates where the data returned for a load operation was sourced. It might also include other information, such as the state of the data at the source. It is IMPLEMENTATION DEFINED and might be UNPREDICTABLE whether this is included for load and atomic operations that generate an external abort. It is IMPLEMENTATION DEFINED whether this is included for atomic operations that do not return data to a PE register. Included for all other load and atomic operations.

Attributes

Multi-part packet comprising:

- 8-bit header.
- 8 or 16-bit payload.

Data Source packet header

The Data Source packet header bit assignments are:

7	6	5	4	3	2	1	0	
0	1	SZ		0	0	1	1	Byte 0

Byte 0 bits [7:6]

This field reads as `0b01`.

SZ, byte 0 bits [5:4]

Payload size. The defined values of this field are:

`0b00`	Byte.
`0b01`	Halfword.

Byte 0 bits [3:0]

This field reads as `0b0011`.

Data Source packet payload

When SZ == `0b00`, the Data Source packet payload bit assignments are:

7	6	5	4	3	2	1	0	
SOURCE								Byte 0

When SZ == `0b01`, the Data Source packet payload bit assignments are:

7	6	5	4	3	2	1	0	
SOURCE[7:0]								Byte 0
SOURCE[15:8]								Byte 1

SOURCE, byte <0>, when SZ == `0b00`

SOURCE, bytes <1:0>, when SZ == `0b01`

Because the list of data sources varies from system to system, the definition of this field is IMPLEMENTATION DEFINED. If a sampled operation generated multiple data accesses, it is IMPLEMENTATION DEFINED how the data source information is combined.

5.3.6 End packet

The End packet characteristics are:

Purpose

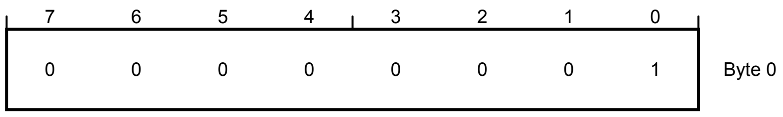
Defines the end of a record if a Timestamp packet is not present.

Attributes

8-bit packet.

Field descriptions

The End packet bit assignments are:



Byte <0>

This field reads as `0b00000001`.

5.3.7 Events packet

The Events packet characteristics are:

Purpose

Indicates up to 64 events generated by the sampled operation.

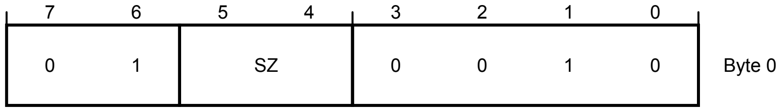
Attributes

Multi-part packet comprising:

- 8-bit header.
- 8, 16, 32, or 64-bit payload.

Events packet header

The Events packet header bit assignments are:



Byte 0 bits [7:6]

This field reads as `0b01`.

SZ, byte 0 bits [5:4]

Payload size. The defined values of this field are:

`0b00`	Byte.
`0b01`	Halfword.
`0b10`	Word.
`0b11`	Doubleword.

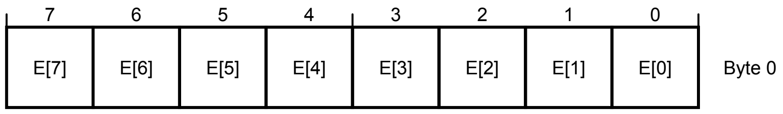
Software must treat bits that are not output as zero.

Byte 0 bits [3:0]

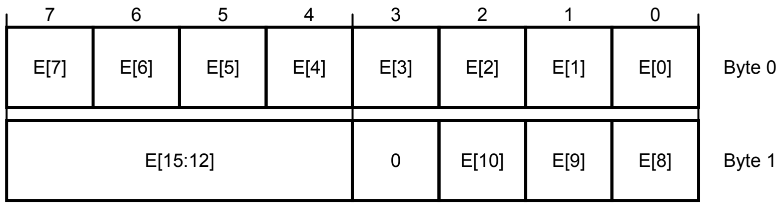
This field reads as `0b0010`.

Events packet payload

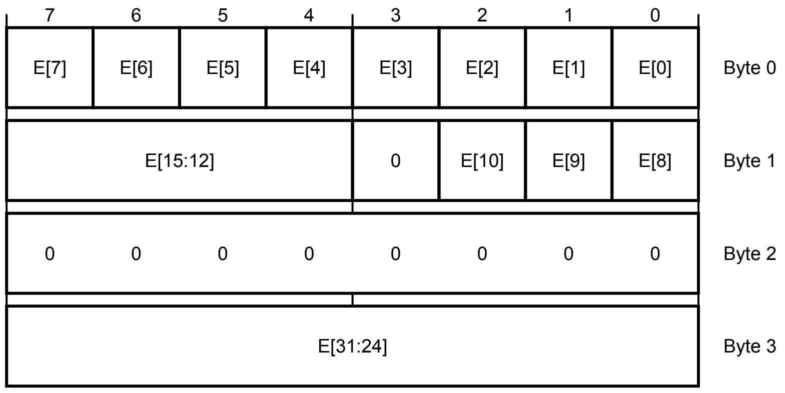
When SZ == `0b00`, the Events packet payload bit assignments are:



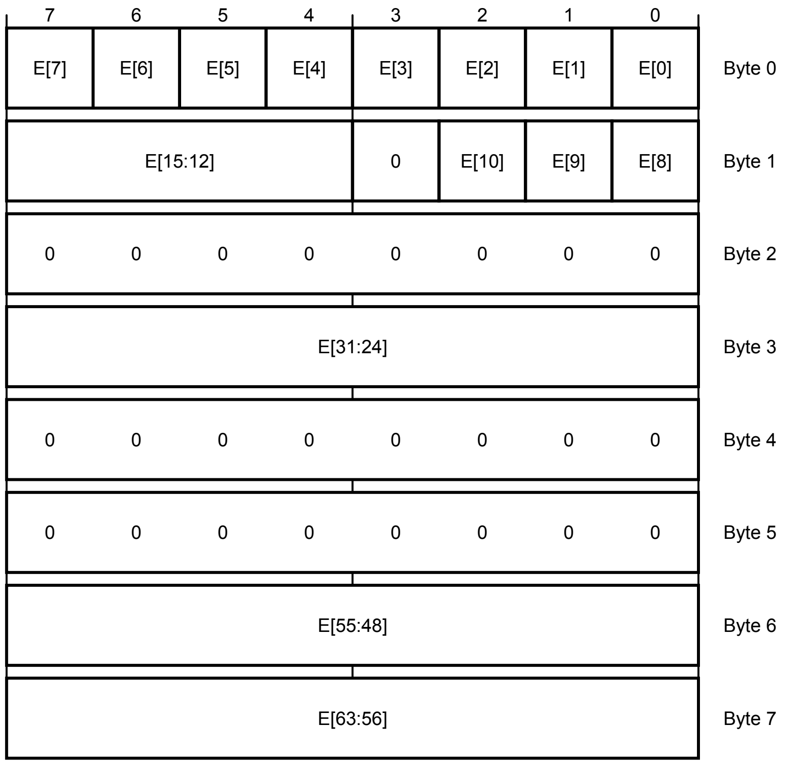
When SZ == `0b01`, the Events packet payload bit assignments are:



When SZ == `0b10`, the Events packet payload bit assignments are:


7 6 5 4 3 2 1 0
E[7] E[6] E[5] E[4] E[3] E[2] E[1] E[0] Byte 0
E[15:12] 0 E[10] E[9] E[8] Byte 1
0 0 0 0 0 0 0 0 Byte 2
E[31:24] Byte 3

When SZ == `0b11`, the Events packet payload bit assignments are:



E[63:48], bytes <7:6>, when SZ == `0b11`

Events 63 to 48. IMPLEMENTATION DEFINED.

Bytes <5:4,2>, byte 1 bit [3], when SZ == `0b11`

This field reads-as-zero.

E[31:24], byte <3>, when SZ == `0b10`, or when SZ == `0b11`

Events 31 to 24. IMPLEMENTATION DEFINED.

Byte <2>, byte 1 bit [3], when SZ == `0b10`

This field reads-as-zero.

E[15:12], byte 1 bits [7:4], when SZ == `0b01`, when SZ == `0b10`, or when SZ == `0b11`

Events 15 to 12. IMPLEMENTATION DEFINED.

Byte 1 bit [3], when SZ == `0b01`

This bit reads-as-zero.

E[10], byte 1 bit [2], when SZ == `0b01`, when SZ == `0b10`, or when SZ == `0b11`

Remote access. The defined values of this bit are:

`0`	Did not cause access to another socket.
`1`	Load/store operation caused an access to another socket in a multi-socket system. This includes each data memory access that accesses another socket in a multi-socket system, including those that do not return data.

Note

See also REMOTE_ACCESS.

E[9], byte 1 bit [1], when SZ == `0b01`, when SZ == `0b10`, or when SZ == `0b11`

Last Level cache miss. The defined values of this bit are:

`0` Did not miss Last Level cache.

`1` Load/store operation caused an access to at least the Last Level cache but is not completed by the Last Level cache. That is, each:

- Load operation that does not return data from the Last Level cache.
- Store operation that does not update the Last Level cache.

The event is not set for operations that are completed by a cache above the Last Level cache.

Note

See also LL_CACHE_MISS.

E[8], byte 1 bit [0], when SZ == `0b01`, when SZ == `0b10`, or when SZ == `0b11`

Last Level cache access. The defined values of this bit are:

`0`	Did not access Last Level data or unified cache.
`1`	Load/store operation caused a cache access to at least the Last Level data or unified cache.

Note

The architecture does not define the Last Level cache. The Last Level cache is typically the largest cache on this device shared by all PEs in the inner or outer Shareable domain of this PE. In a multi-socket system, it is IMPLEMENTATION DEFINED whether this includes caches on other sockets.

See also LL_CACHE.

E[7], byte 0 bit [7]

Mispredicted. The defined values of this bit are:

`0`	Did not cause correction to the predicted program flow.
`1`	A branch that caused a correction to the predicted program flow.

E[6], byte 0 bit [6]

Not taken. The defined values of this bit are:

`0` Did not fail condition code check.

`1` A conditional instruction that failed its condition code check. This includes conditional branches, compare-and-branch, conditional select, and conditional compares:

- For a conditional branch or compare-and-branch instruction, this means the branch was not taken.
- For a conditional select, this means the second operand was written to the result.
- For a condition compare, this means the condition flags were set to the immediate value and not the result of the compare.

E[5], byte 0 bit [5]

TLB walk. The defined values of this bit are:

`0` Did not generate TLB walk.

`1` Load/store operation that causes a refill of a data or unified TLB, involving at least one translation table walk access. This includes each complete or partial translation table walk that causes an access to memory, including to data or translation table walk caches.

Note

See also DTLB_WALK.

E[4], byte 0 bit [4]

TLB access. The defined values of this bit are:

`0` Did not access TLB.

`1` Load/store operation caused an access to at least the first level of data or unified TLB.

Note

See also L1D_TLB.

E[3], byte 0 bit [3]

Level 1 Data cache refill. The defined values of this bit are:

`0` Did not cause level 1 data cache refill.

`1` Load/store operation caused a refill of at least the first level of data or unified cache. This includes each data memory access that causes a refill from outside the cache. It excludes accesses that do not cause a new cache refill but are satisfied from refilling data of a previous miss.

Note

See also L1D_CACHE_REFILL.

E[2], byte 0 bit [2]

Level 1 Data cache access. The defined values of this bit are:

`0` Did not access level 1 data cache.

`1` Load/store operation caused a cache access to at least the first level of data or unified cache.

Note

See also L1D_CACHE.

E[1], byte 0 bit [1]

Architecturally retired. The defined values of this bit are:

`0` Did not retire.

`1` Committed its results to the architectural state of the PE, or completed with a synchronous architectural exception.

Note

A conditional instruction can retire even if it fails its condition code check.

E[0], byte 0 bit [0]

Generated exception. The defined values of this bit are:

`0` Did not generate an exception.

`1` Completed with a synchronous exception.

If E[1] in the same Events packet is set to 0, then the meaning of this bit is IMPLEMENTATION DEFINED.

5.3.8 Operation Type packet

The Operation Type packet characteristics are:

Purpose

Defines the type of operation sampled. Included for all operations.

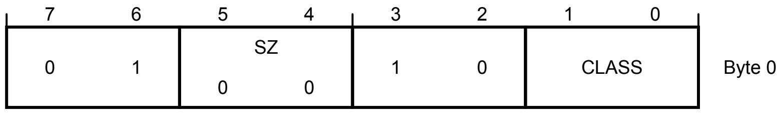
Attributes

Multi-part packet comprising:

- 8-bit header.
- 8-bit payload.

Operation Type packet header

The Operation Type packet header bit assignments are:



Byte 0 bits [7:6]

This field reads as `0b01`.

SZ, byte 0 bits [5:4]

Payload size. The defined values of this field are:

`0b00` Byte.

This field reads as `0b00`.

Byte 0 bits [3:2]

This field reads as `0b10`.

CLASS, byte 0 bits [1:0]

Top-level instruction class. The defined values of this field are:

`0b00` Other.

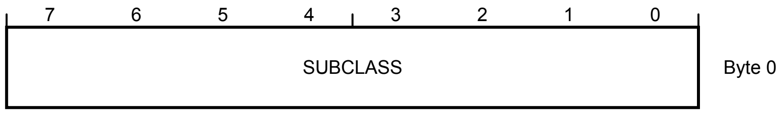
`0b01` Load, store, or atomic.

`0b10` Branch or exception return.

All other values are reserved.

Operation Type packet payload (Other)

The Operation Type packet payload (Other) bit assignments are:



SUBCLASS, byte <0>

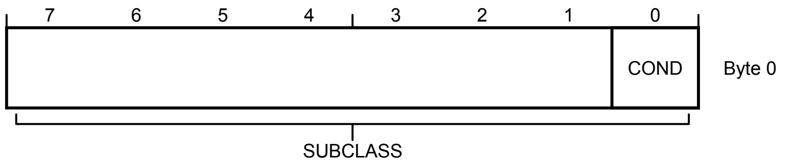
Second-level instruction class. Defines the type of instruction. The defined values of this field are:

`0x00`	Other (unconditional) instruction.
`0x01`	Conditional instruction or select.

All other values are reserved.

Operation Type packet payload (Branch)

The Operation Type packet payload (Branch) bit assignments are:



SUBCLASS, byte <0>

Second-level instruction class. Describes the branch type. The defined values of this field are:

`0b0000000x`	Direct branch.
`0b0000001x`	Indirect branch.

All other values are reserved.

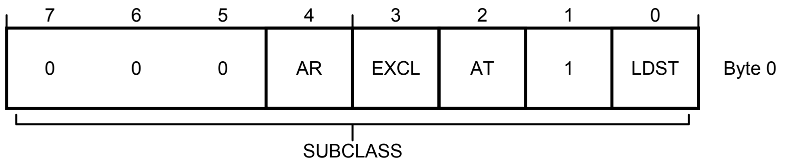
COND, byte 0 bit [0]

Conditional. The defined values of this bit are:

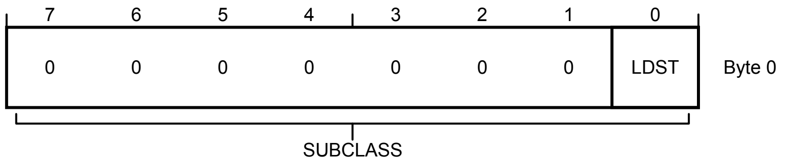
`0`	Unconditional branch.
`1`	Conditional branch.

Operation Type packet payload (Load/store)

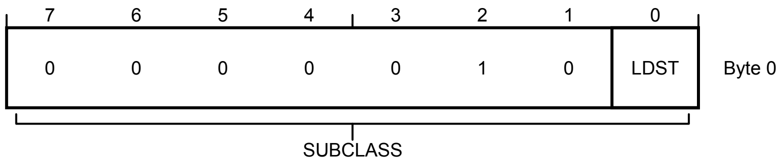
When Extended load/store, the Operation Type packet payload (Load/store) bit assignments are:



When General-purpose load/store, the Operation Type packet payload (Load/store) bit assignments are:



When SIMD&FP load/store, the Operation Type packet payload (Load/store) bit assignments are:



SUBCLASS, byte <0>

Second-level instruction class. Indicates the load/store type. The defined values of this field are:

`0b0000000x`	A load/store targeting the general-purpose registers, other than an atomic operation, load-acquire, store-release or exclusive.
`0b000xxx1x`	An atomic operation, load-acquire, store-release or exclusive. Bits [4:2] are further subdivided as described by the AR, EXCL and AT fields.
`0b0000010x`	A load/store targeting the SIMD&FP registers.

All other values are reserved.

AR, byte 0 bit [4], when Extended load/store

Acquire/Release. The defined values of this bit are:

`0`	Load/store/atomic without Acquire or Release semantics.
`1`	Load/store/atomic with Acquire or Release semantics.

EXCL, byte 0 bit [3], when Extended load/store

Exclusive. The defined values of this bit are:

`0`	Load/store/atomic without Exclusive.
`1`	Load/store with Exclusive.

This bit is RES0 if AT == 1.

AT, byte 0 bit [2], when Extended load/store

Atomic load/store. The defined values of this bit are:

`0`	Not atomic.
`1`	Atomic.

LDST, byte 0 bit [0]

Store not load. The defined values of this bit are:

`0`	Load or swap.
`1`	Store.

5.3.9 Padding

The Padding characteristics are:

Purpose

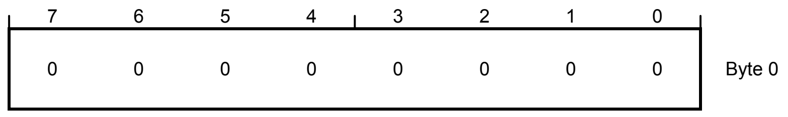
Allows the PE to create alignment in the protocol buffer.

Attributes

8-bit packet.

Field descriptions

The Padding bit assignments are:



Byte <0>

This field reads as `0b00000000`.

5.3.10 Timestamp packet

The Timestamp packet characteristics are:

Purpose

The 64-bit timestamp value when the operation was sampled. The Timestamp packet must come at the end of the record. If the Timestamp packet is not present, an End packet must come at the end of the record.

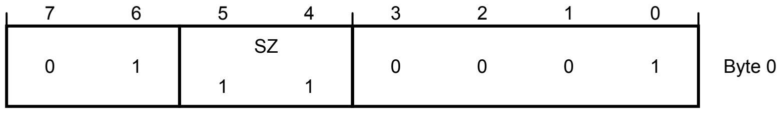
Attributes

Multi-part packet comprising:

- 8-bit header.
- 64-bit payload.

Timestamp packet header

The Timestamp packet header bit assignments are:



Byte 0 bits [7:6]

This field reads as `0b01`.

SZ, byte 0 bits [5:4]

Payload size. The defined values of this field are:

`0b11`	Doubleword.

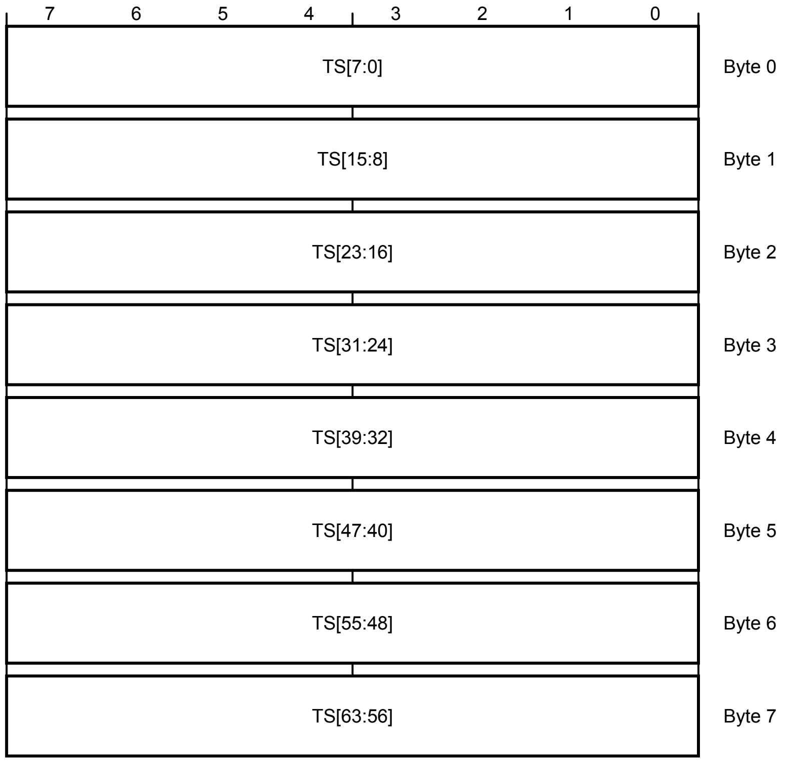
This field reads as `0b11`.

Byte 0 bits [3:0]

This field reads as `0b0001`.

Timestamp packet payload

The Timestamp packet payload bit assignments are:



TS, bytes <7:0>

Timestamp value when the operation was sampled. The value depends on the result of `CollectTimeStamp()`:

- If `TimeStamp_Virtual`, this is the virtual timestamp, CNTVCT_EL0.
- If `TimeStamp_Physical`, this is the physical timestamp, CNTPCT_EL0.
- If `TimeStamp_None`, the timestamp packet is not included and an End packet must come at the end of the record.

However, if the Generic Timer System counter is disabled and `CollectTimeStamp()` returns a value other than `TimeStamp_None`, then it is IMPLEMENTATION DEFINED whether:

- The Statistical Profiling Extension behaves as if `CollectTimeStamp()` returns the value `TimeStamp_None`.
- The value of this field in the record is UNKNOWN.

Note

This relaxation refers to when the actual System counter is disabled, that is, CNTEN.EN == 0. It does not apply when the System counter is enabled but not accessible at the current Exception level.

5.4 IMPLEMENTATION DEFINED packets

The Address packet, Events packet, and Counter packet allow additional IMPLEMENTATION DEFINED data to be recorded. A tool can skip this data if it is not understood. However, an implementation might include IMPLEMENTATION DEFINED controls to enable this data to be recorded.