PMSIDR_EL1, Sampling Profiling ID Register

The PMSIDR_EL1 characteristics are:

Purpose

Describes the Statistical Profiling implementation to software

Configuration

This register is present only when FEAT_SPE is implemented. Otherwise, direct accesses to PMSIDR_EL1 are undefined.

Attributes

PMSIDR_EL1 is a 64-bit register.

Field descriptions

63 62 61 60 59 58	57	56	55 54 53 52	51 50 49 48	47 46 45 44	43 42 41 40	39	38	37	36	35	34	33	32
RES0														
RES0	CRR	PBT	Format	CountSize	MaxSize	Interval	FDS	FnE	ERnd	LDS	ArchInst	FL	FT	FE
31 30 29 28 27 26	25	24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7	6	5	4	3	2	1	0

Bits [63:26]

Reserved, res0.

CRR, bit [25]

Call Return branch records. Defined values are:

CRR	Meaning
`0b0`	Operation Type packets for branches do not contain Call Return information.
`0b1`	Operation Type packets for branches contain Call Return information.

FEAT_SPE_CRR adds the functionality identified by the value 1.

PBT, bit [24]

Previous branch target Address packet. Defined values are:

PBT	Meaning
0b0	Previous branch target Address packet not supported.
0b1	Previous branch target Address packet support implemented.

FEAT_SPEv1p2 adds the optional functionality identified by the value 1.

Format, bits [23:20]

Defines the format of the sample records. Defined values are:

Format	Meaning
0b0000	Format 0.

All other values are reserved.

CountSize, bits [19:16]

Defines the size of the counters.

CountSize	Meaning	Applies when
0b0010	12-bit saturating counters.	
0b0011	16-bit saturating counters.	From Armv8.8

All other values are reserved.

This field has an implementation defined value.

Access to this field is **RO**.

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. Defined values are:

MaxSize	Meaning
0b0100	16 bytes.
0b0101	32 bytes.
0b0110	64 bytes.
0b0111	128 bytes.
0b1000	256 bytes.

0b1001	512 bytes.
0b1010	1KB.
0b1011	2KB.

All other values are reserved.

The values 0b0100 and 0b0101 are not permitted for an implementation.

This field has an implementation defined value.

Access to this field is **RO**.

Interval, bits [11:8]

Recommended minimum sampling interval. This provides guidance from the implementer to the smallest minimum sampling interval, N. Defined values are:

Interval	Meaning
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.

This field has an implementation defined value.

Access to this field is **RO**.

FDS, bit [7]

Filter by data source. Defined values are:

FDS	Meaning
0b0	PMSDSFR_EL1 is not implemented and PMSFCR_EL1.FDS is res0.
0b1	PMSDSFR_EL1 and PMSFCR_EL1.FDS are implemented.

FEAT_SPE_FDS adds the functionality identified by the value 1.

FnE, bit [6]

Filtering by events, inverted. Defined values are:

FnE	Meaning
`0b0`	PMSNEVFR_EL1 is not implemented and PMSFCR_EL1.FnE is res0.
`0b1`	PMSNEVFR_EL1 and PMSFCR_EL1.FnE are implemented.

FEAT_SPEv1p2 adds the functionality identified by the value 1.

ERnd, bit [5]

Defines how the random number generator is used in determining the interval between samples, when enabled by PMSIRR_EL1.RND. Defined values are:

ERnd	Meaning
`0b0`	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.
`0b1`	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.

This field has an implementation defined value.

Access to this field is **RO**.

LDS, bit [4]

Data source indicator for sampled load instructions. Defined values are:

LDS	Meaning
`0b0`	Loaded data source not implemented.
`0b1`	Loaded data source implemented.

This field has an implementation defined value.

Access to this field is **RO**.

ArchInst, bit [3]

Architectural instruction profiling. Defined values are:

ArchInst	Meaning
`0b0`	Micro-op sampling implemented.
`0b1`	Architecture instruction sampling implemented.

This field has an implementation defined value.

Access to this field is **RO**.

FL, bit [2]

Filtering by latency. This bit is RAO.

FT, bit [1]

Filtering by operation type. This bit is RAO.

FE, bit [0]

Filtering by events. This bit is RAO.

Accessing PMSIDR_EL1

Accesses to this register use the following encodings in the System register encoding space:

```
MRS <Xt>, PMSIDR_EL1
```

op0	op1	CRn	CRm	op2
0b11	0b000	0b1001	0b1001	0b111

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' ||
MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE !=
SCR_EL3.NSE)) then
        UNDEFINED;
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.PMSIDR_EL1 ==
'1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMS == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' ||
MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE !=
SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSIDR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && (MDCR_EL3.NSPB[0] == '0' ||
MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE !=
SCR_EL3.NSE)) then
        UNDEFINED;
    elsif HaveEL(EL3) && (MDCR_EL3.NSPB[0] == '0' ||
MDCR_EL3.NSPB[1] != SCR_EL3.NS ||
(IsFeatureImplemented(FEAT_RME) && MDCR_EL3.NSPBE !=
SCR_EL3.NSE)) then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMSIDR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMSIDR_EL1;
```