

## SMCR\_EL2, SME Control Register (EL2)

The SMCR\_EL2 characteristics are:

### Purpose

This register controls aspects of Streaming SVE that are visible at Exception levels EL2, EL1, and EL0.

### Configuration

This register is present only when FEAT\_SME is implemented. Otherwise, direct accesses to SMCR\_EL2 are undefined.

This register has no effect if the PE is not in Streaming SVE mode, or if EL2 is not enabled in the current Security state.

If EL2 is not implemented, this register is res0 from EL3.

### Attributes

SMCR\_EL2 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																															
FA64		EZT0		RES0																RAZ/WI				LEN							
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 [63:32]

Reserved, res0.

#### FA64, bit [31]

When FEAT\_SME\_FA64 is implemented:

Controls whether execution of an A64 instruction is considered legal when the PE is in Streaming SVE mode.

FA64	Meaning
0b0	This control does not cause any instruction to be treated as legal in Streaming SVE mode.

0b1 This control causes all implemented A64 instructions to be treated as legal in Streaming SVE mode at EL2, if they are treated as legal at EL3.

---

Arm recommends that portable SME software should not rely on this optional feature, and that operating systems should provide a means to test for compliance with this recommendation.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

**Otherwise:**

Reserved, res0.

**EZT0, bit [30]**

**When FEAT\_SME2 is implemented:**

Traps execution at EL2, EL1, and EL0 of the LDR, LUTI2, LUTI4, MOVT, STR, and ZERO instructions that access the ZT0 register to EL2, when EL2 is enabled in the current Security state.

The exception is reported using [ESR\\_EL2](#).EC value 0x1D, with an ISS code of 0x0000004, at a lower priority than a trap due to PSTATE.SM or PSTATE.ZA.

---

<b>EZT0</b>	<b>Meaning</b>
0b0	This control causes execution of these instructions at EL2, EL1, and EL0 to be trapped.
0b1	This control does not cause execution of any instruction to be trapped.

---

Changes to this field only affect whether instructions that access ZT0 are trapped. They do not affect the contents of ZT0, which remain valid so long as PSTATE.ZA is 1.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

**Otherwise:**

Reserved, res0.

**Bits [29:9]**

Reserved, res0.

**Bits [8:4]**

Reserved, RAZ/WI.

**LEN, bits [3:0]**

Requests an Effective Streaming SVE vector length (SVL) at EL2 of  $(LEN+1)*128$  bits. This field also defines the Effective Streaming SVE vector length at EL0 when EL2 is implemented and enabled in the current Security state, and  $HCR\_EL2.\{E2H,TGE\}$  is  $\{1,1\}$ .

The Streaming SVE vector length can be any power of two from 128 bits to 2048 bits inclusive. An implementation can support any subset of the architecturally permitted lengths.

When the PE is in Streaming SVE mode, the Effective SVE vector length (VL) is equal to SVL.

When FEAT\_SVE is implemented, and the PE is not in Streaming SVE mode, VL is equal to the Effective Non-streaming SVE vector length. See [ZCR\\_EL2](#).

For all purposes other than returning the result of a direct read of SMCR\_EL2, the PE selects the Effective Streaming SVE vector length by performing checks in the following order:

1. If the requested length is less than the minimum implemented Streaming SVE vector length, then the Effective length is the minimum implemented Streaming SVE vector length.
2. If EL3 is implemented and the requested length is greater than the Effective length at EL3, then the Effective length at EL3 is used.
3. Otherwise, the Effective length is the highest supported Streaming SVE vector length that is less than or equal to the requested length.

An indirect read of SMCR\_EL2.LEN appears to occur in program order relative to a direct write of the same register, without the need for explicit synchronization.

The reset behavior of this field is:

- On a Warm reset, this field resets to an architecturally unknown value.

## Accessing SMCR\_EL2

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic SMCR\_EL2 or SMCR\_EL1 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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

MRS <Xt>, SMCR\_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b110

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
    && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN ==
    'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = SMCR_EL2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = SMCR_EL2;
```

## MSR SMCR\_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b110

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
    && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN ==
'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SMCR_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        SMCR_EL2 = X[t, 64];
```

## MRS <Xt>, SMCR\_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b110

```
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'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elsif CPACR_EL1.SMEN == 'x0' then
```

```

        AArch64.SystemAccessTrap(EL1, 0x1D);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' &&
CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        X[t, 64] = NVMem[0x1F0];
    else
        X[t, 64] = SMCR_EL1;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.ESM == '0' then
        UNDEFINED;
    elseif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN ==
'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elseif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
    elseif HCR_EL2.E2H == '1' then
        X[t, 64] = SMCR_EL2;
    else
        X[t, 64] = SMCR_EL1;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.ESM == '0' then
        AArch64.SystemAccessTrap(EL3, 0x1D);
    else
        X[t, 64] = SMCR_EL1;

```

## MSR SMCR\_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0010	0b110

```

if PSTATE.EL == EL0 then
    UNDEFINED;
elseif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority

```

```

when SDD == '1' && CPTR_EL3.ESM == '0' then
    UNDEFINED;
    elsif CPACR_EL1.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' &&
CPTR_EL2.TSM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.SMEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x1D);
    elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x1D);
        elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
            NVMem[0x1F0] = X[t, 64];
        else
            SMCR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1' && CPTR_EL3.ESM == '0' then
            UNDEFINED;
        elsif HCR_EL2.E2H == '0' && CPTR_EL2.TSM == '1'
then
            AArch64.SystemAccessTrap(EL2, 0x1D);
        elsif HCR_EL2.E2H == '1' && CPTR_EL2.SMEN ==
'x0' then
            AArch64.SystemAccessTrap(EL2, 0x1D);
        elsif HaveEL(EL3) && CPTR_EL3.ESM == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x1D);
        elsif HCR_EL2.E2H == '1' then
            SMCR_EL2 = X[t, 64];
        else
            SMCR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if CPTR_EL3.ESM == '0' then
            AArch64.SystemAccessTrap(EL3, 0x1D);
        else
            SMCR_EL1 = X[t, 64];

```

[AArch32  
Registers](#)

[AArch64  
Registers](#)

[AArch32  
Instructions](#)

[AArch64  
Instructions](#)

[Index by  
Encoding](#)

[External  
Registers](#)

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94

Copyright Â© 2010-2023 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.