

## ZCR\_EL2, SVE Control Register (EL2)

The ZCR\_EL2 characteristics are:

### Purpose

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

### Configuration

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

This register has no effect when EL2 is not enabled in the current Security state, or when FEAT\_SME is implemented and the PE is in Streaming SVE mode.

### Attributes

ZCR\_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																																
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 [63:9]

Reserved, res0.

#### Bits [8:4]

Reserved, RAZ/WI.

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

Requests an Effective Non-streaming SVE vector length at EL2 of  $(LEN+1)*128$  bits. This field also defines the Effective Non-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 Non-streaming SVE vector length can be any power of two from 128 bits to 2048 bits inclusive. An implementation can support a subset of the architecturally permitted lengths. An implementation is required to support all lengths that are powers of two, from 128 bits up to its maximum implemented Non-streaming SVE vector length.

When FEAT\_SME is not implemented, or the PE is not in Streaming SVE mode, the Effective SVE vector length (VL) is equal to the Effective Non-streaming SVE vector length.

When FEAT\_SME is implemented and the PE is in Streaming SVE mode, VL is equal to the Effective Streaming SVE vector length. See [SMCR\\_EL2](#).

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

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

An indirect read of ZCR\_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 ZCR\_EL2

When [HCR\\_EL2.E2H](#) is 1, without explicit synchronization, access from EL2 using the mnemonic ZCR\_EL2 or ZCR\_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>, ZCR\_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b000

```

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.EZ == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0'
then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
    else
        X[t, 64] = ZCR_EL2;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        X[t, 64] = ZCR_EL2;

```

## MSR ZCR\_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0010	0b000

```

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.EZ == '0' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0'

```

```

then
    AArch64.SystemAccessTrap(EL2, 0x19);
elseif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
    if Halted() && EDSCR.SDD == '1' then
        UNDEFINED;
    else
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        ZCR_EL2 = X[t, 64];
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        ZCR_EL2 = X[t, 64];

```

## MRS <Xt>, ZCR\_EL1

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

```

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.EZ == '0' then
        UNDEFINED;
    elseif CPACR_EL1.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x19);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' &&
    CPTR_EL2.TZ == '1' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' &&
    CPTR_EL2.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elseif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
    '111' then
        X[t, 64] = NVMem[0x1E0];
    else
        X[t, 64] = ZCR_EL1;
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
    && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    when SDD == '1'" && CPTR_EL3.EZ == '0' then
        UNDEFINED;
    elseif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x19);

```

```

elseif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0'
then
    AArch64.SystemAccessTrap(EL2, 0x19);
elseif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
    if Halted() && EDSCR.SDD == '1' then
        UNDEFINED;
    else
        AArch64.SystemAccessTrap(EL3, 0x19);
elseif HCR_EL2.E2H == '1' then
    X[t, 64] = ZCR_EL2;
else
    X[t, 64] = ZCR_EL1;
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        X[t, 64] = ZCR_EL1;

```

## MSR ZCR\_EL1, <Xt>

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

```

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.EZ == '0' then
        UNDEFINED;
    elseif CPACR_EL1.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x19);
    elseif EL2Enabled() && HCR_EL2.E2H == '0' &&
CPTR_EL2.TZ == '1' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elseif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.ZEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x19);
    elseif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x19);
    elseif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        NVMem[0x1E0] = X[t, 64];
    else
        ZCR_EL1 = X[t, 64];
elseif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
    && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.EZ == '0' then
        UNDEFINED;

```

```

elseif HCR_EL2.E2H == '0' && CPTR_EL2.TZ == '1'
then
    AArch64.SystemAccessTrap(EL2, 0x19);
elseif HCR_EL2.E2H == '1' && CPTR_EL2.ZEN == 'x0'
then
    AArch64.SystemAccessTrap(EL2, 0x19);
elseif HaveEL(EL3) && CPTR_EL3.EZ == '0' then
    if Halted() && EDSCR.SDD == '1' then
        UNDEFINED;
    else
        AArch64.SystemAccessTrap(EL3, 0x19);
elseif HCR_EL2.E2H == '1' then
    ZCR_EL2 = X[t, 64];
else
    ZCR_EL1 = X[t, 64];
elseif PSTATE.EL == EL3 then
    if CPTR_EL3.EZ == '0' then
        AArch64.SystemAccessTrap(EL3, 0x19);
    else
        ZCR_EL1 = X[t, 64];

```

[AArch32  
Registers](#)

[AArch64  
Registers](#)

[AArch32  
Instructions](#)

[AArch64  
Instructions](#)

[Index by  
Encoding](#)

[External  
Registers](#)

28/03/2023 16:02; 72747e43966d6b97dcbdd230a1b3f0421d1ea3d94

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