SCTLR2_EL2, System Control Register (EL2)

The SCTLR2_EL2 characteristics are:

Purpose

Provides top level control of the system, including its memory system, at EL2.

When FEAT_VHE is implemented and the value of HCR_EL2.{E2H,TGE} is {1,1}, these controls also apply to execution at EL0.

Configuration

This register is present only when FEAT_SCTLR2 is implemented. Otherwise, direct accesses to SCTLR2_EL2 are undefined.

This register has no effect if EL2 is not enabled in the current Security state.

Attributes

SCTLR2_EL2 is a 64-bit register.

Field descriptions

63626160595857565554535251504948474645444342414039	38	37	36	35	34	33
RES0						
RES0	EnIDCP128	EASE	EnANERR	EnADERR	NMEA	EME
313029282726252423222120191817161514131211109 8 7	6	5	4	3	2	1

Bits [63:7]

Reserved, res0.

EnIDCP128, bit [6]
When FEAT_SYSREG128 is implemented:

Enables access to implementation defined 128-bit System registers.

EnIDCP128	Meaning

`0b0`	Accesses at EL0 to implementation defined 128-bit System registers are trapped to EL2 using an ESR_EL2.EC value of `0x14`, unless the access generates a higher priority exception. Disables the functionality of the 128-bit implementation defined System registers that are accessible at EL2.
`0b1`	No accesses are trapped by this control.

This field is ignored by the PE and treated as zero when any of the following are true:

- SCR_EL3.SCTLR2En == 0.
- HCR_EL2.E2H == 0.
- HCR_EL2.TGE == 0.

The reset behavior of this field is:

- On a Warm reset:
  - When EL3 is not implemented, this field resets to `0`.
  - Otherwise, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

EASE, bit [5]
When FEAT_DoubleFault2 is implemented:

External Aborts to SError interrupt exception vector.

EASE	Meaning
`0b0`	Synchronous External Abort exceptions taken to EL2 are taken to the appropriate synchronous exception vector offset from VBAR_EL2.

0b1	Synchronous External Abort exceptions taken to EL2 are taken to the appropriate SError exception vector offset from VBAR_EL2.

This field is ignored by the PE and treated as zero when SCR_EL3.SCTLR2En == 0.

The reset behavior of this field is:

- On a Warm reset:
    - When EL3 is not implemented, this field resets to `0`.
    - Otherwise, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

EnANERR, bit [4]
When FEAT_ANERR is implemented:

Enable Asynchronous Normal Read Error.

EnANERR	Meaning
0b0	External abort on Normal memory reads generate synchronous Data Abort exceptions in the EL2 and EL2&0 translation regimes.
0b1	External abort on Normal memory reads generate synchronous Data Abort or asynchronous SError exceptions in the EL2 and EL2&0 translation regimes.

It is implementation-specific whether this field applies to memory reads generated by each of the following:

- FP&SIMD register loads.
- SVE register loads.
- SME register loads.
- LD<op>, SWP and CAS{P} Atomic instructions that return a value to the PE.
- ST64BV{0} instructions that return a value to the PE.
- RCW instructions that return a value to the PE.

Setting this field to 0 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Normal memory read in every case. There might be implementation-specific circumstances when an error on a load cannot be taken synchronously. These circumstances should be rare enough that treating such occurrences as fatal does not cause a significant increase in failure rate.

Setting this field to 0 might have a performance impact for Normal memory reads.

This field is ignored by the PE and treated as zero when SCR_EL3.SCTLR2En == 0.

The reset behavior of this field is:

- On a Warm reset:
  - When EL3 is not implemented, this field resets to `0`.
  - Otherwise, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

EnADERR, bit [3]
When FEAT_ADERR is implemented:

Enable Asynchronous Device Read Error.

EnADERR	Meaning
`0b0`	External abort on Device memory reads generate synchronous Data Abort exceptions in the EL2 and EL2&0 translation regimes.
`0b1`	External abort on Device memory reads generate synchronous Data Abort or asynchronous SError exceptions in the EL2 and EL2&0 translation regimes.

It is implementation-specific whether this field applies to memory reads generated by each of the following:

- FP&SIMD register loads.
- SVE register loads.
- SME register loads.
- LD<op>, SWP and CAS{P} Atomic instructions that return a value to the PE.
- ST64BV{0} instructions that return a value to the PE.
- RCW instructions that return a value to the PE.

Setting this field to 0 does not guarantee that the PE is able to take a synchronous Data Abort exception for an External abort on a Device memory read in every case. There might be implementation-specific circumstances when an error on a load cannot be taken synchronously. These circumstances should be rare enough that treating such occurrences as fatal does not cause a significant increase in failure rate.

Setting this field to 0 might have a performance impact for Device memory reads.

This field is ignored by the PE and treated as zero when SCR_EL3.SCTLR2En == 0.

The reset behavior of this field is:

- On a Warm reset:
  - When EL3 is not implemented, this field resets to `0`.
  - Otherwise, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

NMEA, bit [2]
When FEAT_DoubleFault2 is implemented:

Non-maskable External Aborts. Controls whether PSTATE.A masks SError exceptions at EL2.

NMEA	Meaning
`0b0`	SError exceptions are not taken at EL2 if PSTATE.A == 1, unless routed to a higher Exception level.
`0b1`	SError exceptions are taken at EL2 regardless of the value of PSTATE.A, unless routed to a higher Exception level.

This field is ignored by the PE and treated as zero when SCR_EL3.SCTLR2En == 0.

The reset behavior of this field is:

- On a Warm reset:
  - When EL3 is not implemented, this field resets to `0`.
  - Otherwise, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

EMEC, bit [1]
When FEAT_MEC is implemented:

Enables MEC for the Realm physical address space at EL2.

EMEC	Meaning
`0b0`	MEC is not enabled for the Realm physical address space at EL2.
`0b1`	MEC is enabled for the Realm physical address space at EL2.

This bit is permitted to be cached in a TLB.

The reset behavior of this field is:

- On a Warm reset:
    - When EL3 is not implemented, this field resets to `0`.
    - Otherwise, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

Bit [0]

Reserved, res0.

Accessing SCTLR2_EL2

When FEAT_VHE is implemented, and HCR_EL2.E2H is 1, without explicit synchronization, accesses from EL2 using the register name SCTLR2_EL2 or SCTLR2_EL1 are not guaranteed to be ordered with respect to accesses using the other register name.

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

```
MRS <Xt>, SCTLR2_EL2
```

op0	op1	CRn	CRm	op2

0b11	0b100	0b0001	0b0000	0b011

```
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'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = SCTLR2_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR2_EL2;
```

MSR SCTLR2_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b0001	0b0000	0b011

```
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'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        SCTLR2_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SCTLR2_EL2 = X[t, 64];
```

MRS <Xt>, SCTLR2_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b011

```
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'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TRVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGRTR_EL2.SCTLR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() ||
HCRX_EL2.SCTLR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        X[t, 64] = NVMem[0x278];
    else
        X[t, 64] = SCTLR2_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        X[t, 64] = SCTLR2_EL2;
    else
        X[t, 64] = SCTLR2_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = SCTLR2_EL1;
```

MSR SCTLR2_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b0001	0b0000	0b011

```
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'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.TVM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HFGWTR_EL2.SCTLR_EL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && (!IsHCRXEL2Enabled() ||
HCRX_EL2.SCTLR2En == '0') then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        NVMem[0x278] = X[t, 64];
    else
        SCTLR2_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'" && SCR_EL3.SCTLR2En == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && SCR_EL3.SCTLR2En == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HCR_EL2.E2H == '1' then
        SCTLR2_EL2 = X[t, 64];
    else
        SCTLR2_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    SCTLR2_EL1 = X[t, 64];
```

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94