ICC_SRE_EL1, Interrupt Controller System Register Enable Register (EL1)

The ICC_SRE_EL1 characteristics are:

Purpose

Controls whether the System register interface or the memory-mapped interface to the GIC CPU interface is used for EL1.

Configuration

This register is banked between ICC_SRE_EL1 and ICC_SRE_EL1_S and ICC_SRE_EL1_NS.

AArch64 System register ICC_SRE_EL1 bits [31:0] (ICC_SRE_EL1_S) are architecturally mapped to AArch32 System register ICC_SRE[31:0] (ICC_SRE_S).

AArch64 System register ICC_SRE_EL1 bits [31:0] (ICC_SRE_EL1_NS) are architecturally mapped to AArch32 System register ICC_SRE[31:0] (ICC_SRE_NS).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_SRE_EL1 are undefined.

Attributes

ICC_SRE_EL1 is a 64-bit register.

This register has the following instances:

- ICC_SRE_EL1, when EL3 is not implemented
- ICC_SRE_EL1_S, when EL3 is implemented
- ICC_SRE_EL1_NS, when EL3 is implemented

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	DIB	DFB	SRE
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 [63:3]

Reserved, res0.

DIB, bit [2]

Disable IRQ bypass.

DIB	Meaning
`0b0`	IRQ bypass enabled.
`0b1`	IRQ bypass disabled.

If EL3 is implemented and GICD_CTLR.DS == 0, this field is a read-only alias of ICC_SRE_EL3.DIB.

If EL3 is implemented and GICD_CTLR.DS == 1, and EL2 is not implemented, this field is a read/write alias of ICC_SRE_EL3.DIB.

If EL3 is not implemented and EL2 is implemented, this field is a read-only alias of ICC_SRE_EL2.DIB.

If GICD_CTLR.DS == 1 and EL2 is implemented, this field is a read-only alias of ICC_SRE_EL2.DIB.

In systems that do not support IRQ bypass, this field is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0`.

DFB, bit [1]

Disable FIQ bypass.

DFB	Meaning
`0b0`	FIQ bypass enabled.
`0b1`	FIQ bypass disabled.

If EL3 is implemented and GICD_CTLR.DS == 0, this field is a read-only alias of ICC_SRE_EL3.DFB.

If EL3 is implemented and GICD_CTLR.DS == 1, and EL2 is not implemented, this field is a read/write alias of ICC_SRE_EL3.DFB.

If EL3 is not implemented and EL2 is implemented, this field is a read-only alias of ICC_SRE_EL2.DFB.

If GICD_CTLR.DS == 1 and EL2 is implemented, this field is a read-only alias of ICC_SRE_EL2.DFB.

In systems that do not support FIQ bypass, this field is RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0`.

SRE, bit [0]

System Register Enable.

SRE	Meaning
`0b0`	The memory-mapped interface must be used. Access at EL1 to any ICC_* System register other than ICC_SRE_EL1 is trapped to EL1.
`0b1`	The System register interface for the current Security state is enabled.

If software changes this bit from 1 to 0 in the Secure instance of this register, the results are unpredictable.

If an implementation supports only a System register interface to the GIC CPU interface, this bit is RAO/WI.

If EL3 is implemented and ICC_SRE_EL3.SRE==0 the Secure copy of this bit is RAZ/WI. If ICC_SRE_EL3.SRE is changed from zero to one, the Secure copy of this bit becomes unknown.

If EL2 is implemented and ICC_SRE_EL2.SRE==0 the Non-secure copy of this bit is RAZ/WI. If ICC_SRE_EL2.SRE is changed from zero to one, the Non-secure copy of this bit becomes unknown.

If EL3 is implemented and ICC_SRE_EL3.SRE==0 the Non-secure copy of this bit is RAZ/WI. If ICC_SRE_EL3.SRE is changed from zero to one, the Non-secure copy of this bit becomes unknown.

If Realm Management Extension is implemented, this field is RAO/WI.

GICv3 implementations that do not require GICv2 compatibility might choose to make this bit RAO/WI. The following options are supported:

- The Non-secure copy of ICC_SRE_EL1.SRE can be RAO/WI if ICC_SRE_EL2.SRE is also RAO/WI. This means all Non-secure software, including VMs using only virtual interrupts, must access the GIC using System registers.
- The Secure copy of ICC_SRE_EL1.SRE can be RAO/WI if ICC_SRE_EL3.SRE and ICC_SRE_EL2.SRE are also RAO/WI. This means that all Secure software must access the GIC using System registers and all Non-secure accesses to registers for physical interrupts must use System registers.

Note

A VM using only virtual interrupts might still use memory-mapped access if the Non-secure copy of ICC_SRE_EL1.SRE is not RAO/WI.

The reset behavior of this field is:

- On a Warm reset, this field resets to `0`.

Accessing ICC_SRE_EL1

Execution with ICC_SRE_EL1.SRE set to 0 might make some System registers unknown.

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

MRS <Xt>, ICC_SRE_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b101

```
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'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ICC_SRE_EL2.Enable == '0'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0'
then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_SRE_EL1_S;
        else
            X[t, 64] = ICC_SRE_EL1_NS;
    else
        X[t, 64] = ICC_SRE_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0'
```

```
then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_SRE_EL1_S;
        else
            X[t, 64] = ICC_SRE_EL1_NS;
    else
        X[t, 64] = ICC_SRE_EL1;
elsif PSTATE.EL == EL3 then
    if SCR_EL3.NS == '0' then
        X[t, 64] = ICC_SRE_EL1_S;
    else
        X[t, 64] = ICC_SRE_EL1_NS;
```

MSR ICC_SRE_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	0b101

```
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'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif EL2Enabled() && ICC_SRE_EL2.Enable == '0'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0'
then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_SRE_EL1_S = X[t, 64];
        else
            ICC_SRE_EL1_NS = X[t, 64];
    else
        ICC_SRE_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'" && ICC_SRE_EL3.Enable == '0' then
        UNDEFINED;
    elsif HaveEL(EL3) && ICC_SRE_EL3.Enable == '0'
then
```

```
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) then
            if SCR_EL3.NS == '0' then
                ICC_SRE_EL1_S = X[t, 64];
            else
                ICC_SRE_EL1_NS = X[t, 64];
        else
            ICC_SRE_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if SCR_EL3.NS == '0' then
            ICC_SRE_EL1_S = X[t, 64];
        else
            ICC_SRE_EL1_NS = X[t, 64];
```

28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94