# ICC_CTLR_EL1, Interrupt Controller Control Register (EL1)

The ICC_CTLR_EL1 characteristics are:

## Purpose

Controls aspects of the behavior of the GIC CPU interface and provides information about the features implemented.

## Configuration

This register is banked between ICC_CTLR_EL1 and ICC_CTLR_EL1_S and ICC_CTLR_EL1_NS.

AArch64 System register ICC_CTLR_EL1 bits [31:0] (ICC_CTLR_EL1_S) are architecturally mapped to AArch32 System register ICC_CTLR[31:0] (ICC_CTLR_S).

AArch64 System register ICC_CTLR_EL1 bits [31:0] (ICC_CTLR_EL1_NS) are architecturally mapped to AArch32 System register ICC_CTLR[31:0] (ICC_CTLR_NS).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_CTLR_EL1 are undefined.

## Attributes

ICC_CTLR_EL1 is a 64-bit register.

This register has the following instances:

- ICC_CTLR_EL1, when EL3 is not implemented
- ICC_CTLR_EL1_S, when EL3 is implemented
- ICC_CTLR_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 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

| 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 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RES0 | | | | | | | | | | | | ExtRange | RSS | RES0 | | A3V | SEIS | IDbits | | | PRIbits | | | RES0 | PMHE | RES0 | | | | EOImode | CBP |

**Bits [63:20]**

Reserved, res0.

**ExtRange, bit [19]**

Extended INTID range (read-only).

| ExtRange | Meaning |
|---|---|
| 0b0 | CPU interface does not support INTIDs in the range 1024..8191.<br>• Behavior is unpredictable if the IRI delivers an interrupt in the range 1024 to 8191 to the CPU interface.<br>**Note**<br>Arm strongly recommends that the IRI is not configured to deliver interrupts in this range to a PE that does not support them. |
| 0b1 | CPU interface supports INTIDs in the range 1024..8191<br>• All INTIDs in the range 1024..8191 are treated as requiring deactivation. |

If EL3 is implemented, ICC_CTLR_EL1.ExtRange is an alias of ICC_CTLR_EL3.ExtRange.

**RSS, bit [18]**

Range Selector Support. Possible values are:

| RSS | Meaning |
|---|---|
| 0b0 | Targeted SGIs with affinity level 0 values of 0 - 15 are supported. |
| 0b1 | Targeted SGIs with affinity level 0 values of 0 - 255 are supported. |

This bit is read-only.

**Bits [17:16]**

Reserved, res0.

**A3V, bit [15]**

Affinity 3 Valid. Read-only and writes are ignored. Possible values are:

| A3V | Meaning |
|---|---|
| 0b0 | The CPU interface logic only supports zero values of Affinity 3 in SGI generation System registers. |
| 0b1 | The CPU interface logic supports nonzero values of Affinity 3 in SGI generation System registers. |

If EL3 is implemented, this bit is an alias of ICC_CTLR_EL3.A3V.

**SEIS, bit [14]**

SEI Support. Read-only and writes are ignored. Indicates whether the CPU interface supports local generation of SEIs:

| SEIS | Meaning |
|---|---|
| 0b0 | The CPU interface logic does not support local generation of SEIs. |
| 0b1 | The CPU interface logic supports local generation of SEIs. |

If EL3 is implemented, this bit is an alias of ICC_CTLR_EL3.SEIS.

**IDbits, bits [13:11]**

Identifier bits. Read-only and writes are ignored. The number of physical interrupt identifier bits supported:

| IDbits | Meaning |
|---|---|
| 0b000 | 16 bits. |
| 0b001 | 24 bits. |

All other values are reserved.

If EL3 is implemented, this field is an alias of ICC_CTLR_EL3.IDbits.

**PRIbits, bits [10:8]**

Priority bits. Read-only and writes are ignored. The number of priority bits implemented, minus one.

An implementation that supports two Security states must implement at least 32 levels of physical priority (5 priority bits).

An implementation that supports only a single Security state must implement at least 16 levels of physical priority (4 priority bits).

---

**Note**

This field always returns the number of priority bits implemented, regardless of the Security state of the access or the value of GICD_CTLR.DS.

---

For physical accesses, this field determines the minimum value of ICC_BPR0_EL1.

If EL3 is implemented, physical accesses return the value from ICC_CTLR_EL3.PRIbits.

If EL3 is not implemented, physical accesses return the value from this field.

**Bit [7]**

Reserved, res0.

**PMHE, bit [6]**

Priority Mask Hint Enable. Controls whether the priority mask register is used as a hint for interrupt distribution:

| PMHE | Meaning |
|---|---|
| 0b0 | Disables use of ICC_PMR_EL1 as a hint for interrupt distribution. |
| 0b1 | Enables use of ICC_PMR_EL1 as a hint for interrupt distribution. |

If EL3 is implemented, this bit is an alias of ICC_CTLR_EL3.PMHE. Whether this bit can be written as part of an access to this register depends on the value of GICD_CTLR.DS:

- If GICD_CTLR.DS == 0, this bit is read-only.
- If GICD_CTLR.DS == 1, this bit is read/write.

If EL3 is not implemented, it is implementation defined whether this bit is read-only or read/write:

- If this bit is read-only, an implementation can choose to make this field RAZ/WI or RAO/WI.

- If this bit is read/write, it resets to zero.

**Bits [5:2]**

Reserved, res0.

**EOImode, bit [1]**

EOI mode for the current Security state. Controls whether a write to an End of Interrupt register also deactivates the interrupt:

| EOImode | Meaning |
|---|---|
| 0b0 | ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide both priority drop and interrupt deactivation functionality. Accesses to ICC_DIR_EL1 are unpredictable. |
| 0b1 | ICC_EOIR0_EL1 and ICC_EOIR1_EL1 provide priority drop functionality only. ICC_DIR_EL1 provides interrupt deactivation functionality. |

The Secure ICC_CTLR_EL1.EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1S.

The Non-secure ICC_CTLR_EL1.EOImode is an alias of ICC_CTLR_EL3.EOImode_EL1NS

**CBPR, bit [0]**

Common Binary Point Register. Controls whether the same register is used for interrupt preemption of both Group 0 and Group 1 interrupts:

| CBPR | Meaning |
|---|---|
| 0b0 | ICC_BPR0_EL1 determines the preemption group for Group 0 interrupts only. ICC_BPR1_EL1 determines the preemption group for Group 1 interrupts. |
| 0b1 | ICC_BPR0_EL1 determines the preemption group for both Group 0 and Group 1 interrupts. |

If EL3 is implemented:

- This bit is an alias of ICC_CTLR_EL3.CBPR_EL1{S,NS} where S or NS corresponds to the current Security state.
- If GICD_CTLR.DS == 0, this bit is read-only.
- If GICD_CTLR.DS == 1, this bit is read/write.

If EL3 is not implemented, this bit is read/write.

The reset behavior of this field is:

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

## Accessing ICC_CTLR_EL1

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

### MRS <Xt>, ICC_CTLR_EL1

| op0 | op1 | CRn | CRm | op2 |
|---|---|---|---|---|
| 0b11 | 0b000 | 0b1100 | 0b1100 | 0b100 |

```
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.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_CTLR_EL1;
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11'
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_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
    else
        X[t, 64] = ICC_CTLR_EL1;
```

```
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && SCR_EL3.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11'
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_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
    else
        X[t, 64] = ICC_CTLR_EL1;
elsif PSTATE.EL == EL3 then
    if ICC_SRE_EL3.SRE == '0' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_CTLR_EL1_S;
        else
            X[t, 64] = ICC_CTLR_EL1_NS;
```

# MSR ICC_CTLR_EL1, <Xt>

| op0 | op1 | CRn | CRm | op2 |
|---|---|---|---|---|
| 0b11 | 0b000 | 0b1100 | 0b1100 | 0b100 |

```
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.<IRQ,FIQ> == '11' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.FMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_CTLR_EL1 = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11'
then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
```

```
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) then
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t, 64];
            else
                ICC_CTLR_EL1_NS = X[t, 64];
        else
            ICC_CTLR_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.<IRQ,FIQ> == '11' then
            UNDEFINED;
        elsif ICC_SRE_EL2.SRE == '0' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && SCR_EL3.<IRQ,FIQ> == '11'
    then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif HaveEL(EL3) then
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t, 64];
            else
                ICC_CTLR_EL1_NS = X[t, 64];
        else
            ICC_CTLR_EL1 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        if ICC_SRE_EL3.SRE == '0' then
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            if SCR_EL3.NS == '0' then
                ICC_CTLR_EL1_S = X[t, 64];
            else
                ICC_CTLR_EL1_NS = X[t, 64];
```



28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94