ICC_BPR1_EL1, Interrupt Controller Binary Point Register 1

The ICC_BPR1_EL1 characteristics are:

Purpose

Defines the point at which the priority value fields split into two parts, the group priority field and the subpriority field. The group priority field determines Group 1 interrupt preemption.

Configuration

This register is banked between ICC_BPR1_EL1 and ICC_BPR1_EL1_S and ICC_BPR1_EL1_NS.

AArch64 System register ICC_BPR1_EL1 bits [31:0] (ICC_BPR1_EL1_S) are architecturally mapped to AArch32 System register ICC_BPR1[31:0] (ICC_BPR1_S).

AArch64 System register ICC_BPR1_EL1 bits [31:0] (ICC_BPR1_EL1_NS) are architecturally mapped to AArch32 System register ICC_BPR1[31:0] (ICC_BPR1_NS).

This register is present only when FEAT_GICv3 is implemented. Otherwise, direct accesses to ICC_BPR1_EL1 are undefined.

Virtual accesses to this register update ICH_VMCR_EL2.VBPR1.

Attributes

ICC_BPR1_EL1 is a 64-bit register.

This register has the following instances:

- ICC_BPR1_EL1, when EL3 is not implemented
- ICC_BPR1_EL1_S, when EL3 is implemented
- ICC_BPR1_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	BinaryPoint

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.

BinaryPoint, bits [2:0]

If the GIC is configured to use separate binary point fields for Group 0 and Group 1 interrupts, the value of this field controls how the 8-bit interrupt priority field is split into a group priority field, that determines interrupt preemption, and a subpriority field. For more information about priorities, see 'Priority grouping' in ARMÂ® Generic Interrupt Controller Architecture Specification, GIC architecture version 3.0 and version 4.0 (ARM IHI 0069).

The minimum value of the Non-secure copy of this register is the minimum value of ICC_BPR0_EL1 + 1. The minimum value of the Secure copy of this register is the minimum value of ICC_BPR0_EL1.

If EL3 is implemented and ICC_CTLR_EL3.CBPR_EL1S is 1:

- Accesses to this register from Secure EL2 access the state of ICC_BPR0_EL1.
- Accesses to this register from Secure EL1:
  - When SCR_EL3.EEL2 is 1 and HCR_EL2.IMO is 1, access the state of ICV_BPR1_EL1.
  - Otherwise, access the state of ICC_BPR0_EL1.

If EL3 is implemented and ICC_CTLR_EL3.CBPR_EL1NS is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of HCR_EL2.IMO and SCR_EL3.IRQ:

HCR_EL2.IMO	SCR_EL3.IRQ	Behavior
`0b0`	`0b0`	Non-secure EL1 and EL2 reads return ICC_BPR0_EL1 + 1 saturated toÂ `0b111`. Non-secure EL1 and EL2 writes are ignored.
`0b0`	`0b1`	Non-secure EL1 and EL2 accesses trap to EL3.

HCR_EL2.IMO	SCR_EL3.IRQ	Behavior
`0b1`	`0b0`	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return ICC_BPR0_EL1 + 1 saturated toÂ `0b111`. Non-secure EL2 writes are ignored.
`0b1`	`0b1`	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 accesses trap to EL3.

If EL3 is not implemented and ICC_CTLR_EL1.CBPR is 1, Non-secure accesses to this register at EL1 or EL2 behave as follows, depending on the values of HCR_EL2.IMO:

HCR_EL2.IMO	Behavior
`0b0`	Non-secure EL1 and EL2 reads return ICC_BPR0_EL1 + 1 saturated toÂ `0b111`. Non-secure EL1 and EL2 writes are ignored.
`0b1`	Non-secure EL1 accesses affect virtual interrupts. Non-secure EL2 reads return ICC_BPR0_EL1 + 1 saturated toÂ `0b111`. Non-secure EL2 writes are ignored.

The reset behavior of this field is:

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

Accessing ICC_BPR1_EL1

On a reset, the binary point field is unknown.

An attempt to program the binary point field to a value less than the minimum value sets the field to the minimum value.

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

MRS <Xt>, ICC_BPR1_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	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.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_BPR1_EL1;
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' 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_BPR1_EL1_S;
        else
            X[t, 64] = ICC_BPR1_EL1_NS;
    else
        X[t, 64] = ICC_BPR1_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 == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' 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_BPR1_EL1_S;
        else
            X[t, 64] = ICC_BPR1_EL1_NS;
    else
        X[t, 64] = ICC_BPR1_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_BPR1_EL1_S;
        else
            X[t, 64] = ICC_BPR1_EL1_NS;
```

MSR ICC_BPR1_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1100	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.IRQ == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.IMO == '1' then
        ICV_BPR1_EL1 = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_BPR1_EL1_S = X[t, 64];
        else
            ICC_BPR1_EL1_NS = X[t, 64];
    else
        ICC_BPR1_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 == '1' then
        UNDEFINED;
    elsif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_BPR1_EL1_S = X[t, 64];
        else
            ICC_BPR1_EL1_NS = X[t, 64];
    else
        ICC_BPR1_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_BPR1_EL1_S = X[t, 64];
        else
            ICC_BPR1_EL1_NS = X[t, 64];
```

AArch32 Registers | AArch64 Registers | AArch32 Instructions | AArch64 Instructions | Index by Encoding | External Registers

28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94