

ICV_AP1R<n>_EL1, Interrupt Controller Virtual Active Priorities Group 1 Registers, n = 0 - 3

The ICV_AP1R<n>_EL1 characteristics are:

Purpose

Provides information about virtual Group 1 active priorities.

Configuration

AArch64 System register ICV_AP1R<n>_EL1 bits [31:0] are architecturally mapped to AArch32 System register [ICV_AP1R<n>\[31:0\]](#).

This register is present only when FEAT_GICv3 is implemented and EL2 is implemented. Otherwise, direct accesses to ICV_AP1R<n>_EL1 are undefined.

Attributes

ICV_AP1R<n>_EL1 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
NMI	RES0																														
IMPLEMENTATION DEFINED																															
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

NMI, bit [63]

When FEAT_GICv3_NMI is implemented and n == 0:

Indicates whether the running priority is from a NMI.

NMI	Meaning
0b0	There is no active Group 1 NMI, or all active Group 1 NMIs have undergone priority-drop.
0b1	There is an active Group 1 NMI.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

Bits [62:32]

Reserved, res0.

IMPLEMENTATION DEFINED, bits [31:0]

implementation defined.

The reset behavior of this field is:

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

The contents of these registers are implementation defined with the one architectural requirement that the value 0x00000000 is consistent with no interrupts being active.

Accessing ICV_AP1R<n>_EL1

Writing to these registers with any value other than the last read value of the register (or 0x00000000 when there are no Group 1 active priorities) might result in unpredictable behavior of the virtual interrupt prioritization system, causing:

- Interrupts that should preempt execution to not preempt execution.
- Interrupts that should not preempt execution to preempt execution.

ICV_AP1R1_EL1 is only implemented in implementations that support 6 or more bits of priority. ICV_AP1R2_EL1 and ICV_AP1R3_EL1 are only implemented in implementations that support 7 bits of priority. Unimplemented registers are undefined.

Writing to the active priority registers in any order other than the following order might result in unpredictable behavior of the interrupt prioritization system:

- [ICV_AP0R<n>_EL1](#).
- ICV_AP1R<n>_EL1.

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

MRS <Xt>, ICC_AP1R<m>_EL1 ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b0:m[1:0]

```

integer m = UInt(op2<1:0>);

if m == 1 && NUM_GIC_PRIORITY_BITS < 6 then
    UNDEFINED;
elseif (m == 2 || m == 3) && NUM_GIC_PRIORITY_BITS < 7 then
    UNDEFINED;
elseif 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'" && SCR_EL3.IRQ == '1' then
        UNDEFINED;
    elseif ICC_SRE_EL1.SRE == '0' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elseif EL2Enabled() && ICH_HCR_EL2.TALL1 == '1'
    then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif EL2Enabled() && HCR_EL2.IMO == '1' then
        X[t, 64] = ICV_AP1R_EL1[m];
    elseif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[m];
        else
            X[t, 64] = ICC_AP1R_EL1_NS[m];
    else
        X[t, 64] = ICC_AP1R_EL1[m];
elseif 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;
    elseif ICC_SRE_EL2.SRE == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elseif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elseif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            X[t, 64] = ICC_AP1R_EL1_S[m];
        else

```

```

        X[t, 64] = ICC_AP1R_EL1_NS[m];
    else
        X[t, 64] = ICC_AP1R_EL1[m];
    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_AP1R_EL1_S[m];
            else
                X[t, 64] = ICC_AP1R_EL1_NS[m];
            end
        end
    end
end

```

MSR ICC_AP1R<m>_EL1, <Xt> ; Where m = 0-3

op0	op1	CRn	CRm	op2
0b11	0b000	0b1100	0b1001	0b0:m[1:0]

```

integer m = UInt(op2<1:0>);

if m == 1 && NUM_GIC_PRIORITY_BITS < 6 then
    UNDEFINED;
elsif (m == 2 || m == 3) && NUM_GIC_PRIORITY_BITS < 7 then
    UNDEFINED;
elsif 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_AP1R_EL1[m] = X[t, 64];
    elsif HaveEL(EL3) && SCR_EL3.IRQ == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        end
    elsif HaveEL(EL3) then
        if SCR_EL3.NS == '0' then
            ICC_AP1R_EL1_S[m] = X[t, 64];
        else
            ICC_AP1R_EL1_NS[m] = X[t, 64];
        end
    else
        ICC_AP1R_EL1[m] = X[t, 64];
    end
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_AP1R_EL1_S[m] = X[t, 64];
                else
                    ICC_AP1R_EL1_NS[m] = X[t, 64];
            else
                ICC_AP1R_EL1[m] = 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_AP1R_EL1_S[m] = X[t, 64];
                else
                    ICC_AP1R_EL1_NS[m] = X[t, 64];

```

[AArch32
Registers](#)

[AArch64
Registers](#)

[AArch32
Instructions](#)

[AArch64
Instructions](#)

[Index by
Encoding](#)

[External
Registers](#)

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94

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