PMCR_EL0, Performance Monitors Control Register

The PMCR_EL0 characteristics are:

Purpose

Provides details of the Performance Monitors implementation, including the number of counters implemented, and configures and controls the counters.

Configuration

AArch64 System register PMCR_EL0 bits [31:0] are architecturally mapped to AArch32 System register PMCR[31:0].

AArch64 System register PMCR_EL0 bits [7:0] are architecturally mapped to External register PMU.PMCR_EL0[7:0].

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMCR_EL0 are undefined.

Attributes

PMCR_EL0 is a 64-bit register.

Field descriptions

63:33	32
RES0	FZS

IMP	IDCODE	N	RES0	FZO	RES0	LP	LC	DP	X	D	C	P	E
31:24	23:16	15:11	10	9	8	7	6	5	4	3	2	1	0

Bits [63:33]

Reserved, res0.

FZS, bit [32]
When FEAT_SPEv1p2 is implemented:

Freeze-on-SPE event.

Stop counters when PMBLIMITR_EL1.{PMFZ,E} == {1,1} and PMBSR_EL1.S == 1.

In the description of this field:

- If EL2 is implemented, then PMN is MDCR_EL2.HPMN.
- If EL2 is not implemented, then PMN is PMCR_EL0.N.

FZS	Meaning
`0b0`	Do not freeze on Statistical Profiling Buffer Management event.
`0b1`	Affected counters do not count following a Statistical Profiling Buffer Management event.

The counters affected by this field are:

- If EL2 is implemented, event counters PMEVCNTR<n>_EL0 for values of n less than PMN. This applies even when EL2 is disabled in the current Security state.
- If EL2 is not implemented, all event counters PMEVCNTR<n>_EL0.
- If FEAT_PMUv3_ICNTR is implemented, the instruction counter PMICNTR_EL0.

Other event counters and PMCCNTR_EL0 are not affected by this field.

The reset behavior of this field is:

- On a Warm reset:
  - When AArch32 is supported, this field resets to `0`.
  - Otherwise, this field resets to an architecturally unknown value.

Otherwise:

Reserved, res0.

IMP, bits [31:24]
When FEAT_PMUv3p7 is not implemented:

Implementer code.

If this field is zero, then PMCR_EL0.IDCODE is res0 and software must use MIDR_EL1 to identify the PE.

Otherwise, this field and PMCR_EL0.IDCODE identify the PMU implementation to software. The implementer codes are allocated by Arm. A nonzero value has the same interpretation as MIDR_EL1.Implementer.

Use of this field is deprecated.

This field has an implementation defined value.

Access to this field is **RO**.

Otherwise:

Reserved, RAZ.

IDCODE, bits [23:16]
When PMCR_EL0.IMP != 0b00000000:

Identification code. Use of this field is deprecated.

Each implementer must maintain a list of identification codes that are specific to the implementer. A specific implementation is identified by the combination of the implementer code and the identification code.

This field has an implementation defined value.

Access to this field is **RO**.

Otherwise:

Reserved, res0.

N, bits [15:11]

Indicates the number of event counters implemented. This value is in the range of `0b00000`-`0b11111`. If the value is `0b00000`, then only PMCCNTR_EL0 is implemented. If the value is `0b11111`, then PMCCNTR_EL0 and 31 event counters are implemented.

When EL2 is implemented and enabled for the current Security state, reads of this field from EL1 and EL0 return the value of MDCR_EL2.HPMN.

This field has an implementation defined value.

Access to this field is **RO**.

Bit [10]

Reserved, res0.

FZO, bit [9]
When FEAT_PMUv3p7 is implemented:

Freeze-on-overflow.

Stop event counters on overflow.

In the description of this field:

- If EL2 is implemented, then PMN is MDCR_EL2.HPMN.
- If EL2 is not implemented, then PMN is PMCR_EL0.N.

FZO	Meaning
0b0	Do not freeze on overflow.
0b1	Affected counters do not count when any of the following applies: • PMOVSCLR_EL0[(PMN-1):0] is nonzero. • FEAT_PMUv3_ICNTR is implemented and PMOVSCLR_EL0.F0 is nonzero.

The counters affected by this field are:

- If EL2 is implemented, event counters PMEVCNTR<n>_EL0 for values of n less than PMN. This applies even when EL2 is disabled in the current Security state.
- If EL2 is not implemented, all event counters PMEVCNTR<n>_EL0.
- If FEAT_PMUv3_ICNTR is implemented, the instruction counter PMICNTR_EL0.

Other event counters and PMCCNTR_EL0 are not affected by this field.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

Bit [8]

Reserved, res0.

LP, bit [7]
When FEAT_PMUv3p5 is implemented:

Long event counter enable.

Determines which event counter bit generates an overflow recorded by PMOVSR[n].

In the description of this field:

- If EL2 is implemented, then PMN is MDCR_EL2.HPMN.
- If EL2 is not implemented, then PMN is PMCR_EL0.N.

LP	Meaning
`0b0`	Affected counters overflow on unsigned overflow of PMEVCNTR<n>_EL0[31:0].
`0b1`	Affected counters overflow on unsigned overflow of PMEVCNTR<n>_EL0[63:0].

The counters affected by this field are:

- If EL2 is implemented, event counters PMEVCNTR<n>_EL0 for values of n less than PMN. This applies even when EL2 is disabled in the current Security state.
- If EL2 is not implemented, all event counters PMEVCNTR<n>_EL0.

Other event counters, PMCCNTR_EL0, and, if FEAT_PMUv3_ICNTR is implemented, PMICNTR_EL0 are not affected by this field.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

LC, bit [6]
When AArch32 is supported:

Long cycle counter enable. Determines when unsigned overflow is recorded by the cycle counter overflow bit.

LC	Meaning
`0b0`	Cycle counter overflow on increment that causes unsigned overflow of PMCCNTR_EL0[31:0].
`0b1`	Cycle counter overflow on increment that causes unsigned overflow of PMCCNTR_EL0[63:0].

Arm deprecates use of PMCR_EL0.LC = 0.

The reset behavior of this field is:

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

Otherwise:

Reserved, res1.

DP, bit [5]
When (FEAT_PMUv3p1 is implemented and EL2 is implemented) or EL3 is implemented:

Disable cycle counter when event counting is prohibited.

DP	Meaning
`0b0`	Cycle counting by PMCCNTR_EL0 is not affected by this mechanism.

`0b1`	Cycle counting by PMCCNTR_EL0 is disabled in prohibited regions and when event counting is frozen: • If FEAT_PMUv3p1 is implemented, EL2 is implemented, and MDCR_EL2.HPMD is 1, then cycle counting by PMCCNTR_EL0 is disabled at EL2. • If FEAT_PMUv3p7 is implemented, EL3 is implemented and using AArch64, and MDCR_EL3.MPMX is 1, then cycle counting by PMCCNTR_EL0 is disabled at EL3. • If FEAT_PMUv3p7 is implemented, and event counting is frozen by PMCR_EL0.FZO, then cycle counting by PMCCNTR_EL0 is disabled. • If EL3 is implemented, MDCR_EL3.SPME is 0, and either FEAT_PMUv3p7 is not implemented or MDCR_EL3.MPMX is 0, then cycle counting by PMCCNTR_EL0 is disabled at EL3 and in Secure state.

The conditions when this field disables the cycle counter are the same as when event counting by an event counter PMEVCNTR<n>_EL0 is prohibited or frozen, when either EL2 is not implemented or n is less than MDCR_EL2.HPMN.

For more information, see 'Prohibiting event and cycle counting'.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

X, bit [4]
When the implementation includes a PMU event export bus:

Enable export of events in an implementation defined PMU event export bus.

X	Meaning
`0b0`	Do not export events.
`0b1`	Export events where not prohibited.

This field enables the exporting of events over an implementation defined PMU event export bus to another device, for example to an optional trace unit.

No events are exported when counting is prohibited.

This field does not affect the generation of Performance Monitors overflow interrupt requests or signaling to a cross-trigger interface (CTI) that can be implemented as signals exported from the PE.

The reset behavior of this field is:

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

Otherwise:

Reserved, RAZ/WI.

D, bit [3]
When AArch32 is supported:

Clock divider.

D	Meaning
`0b0`	When enabled, PMCCNTR_EL0 counts every clock cycle.
`0b1`	When enabled, PMCCNTR_EL0 counts once every 64 clock cycles.

If PMCR_EL0.LC == 1, this bit is ignored and the cycle counter counts every clock cycle.

Arm deprecates use of PMCR_EL0.D = 1.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

C, bit [2]

Cycle counter reset. The effects of writing to this bit are:

C	Meaning
`0b0`	No action.
`0b1`	Reset PMCCNTR_EL0 to zero.

Note

Resetting PMCCNTR_EL0 does not change the cycle counter overflow bit. If FEAT_PMUv3p5 is implemented, the value of PMCR_EL0.LC is ignored, and bits [63:0] of the cycle counter are reset.

Access to this field is **WO/RAZ**.

P, bit [1]

Event counter reset.

In the description of this field:

- If EL2 is implemented and is using AArch32, PMN is HDCR.HPMN.
- If EL2 is implemented and is using AArch64, PMN is MDCR_EL2.HPMN.
- If EL2 is not implemented, PMN is PMCR_EL0.N.

P	Meaning
`0b0`	No action.
`0b1`	If n is in the range of affected event counters, resets each event counter PMEVCNTR<n>_EL0 to zero.

The effects of writing to this bit are:

- If EL2 is implemented and enabled in the current Security state, in EL0 and EL1, if PMN is not 0, a write of 1 to this bit resets event counters in the range [0 .. (PMN-1)].
- If EL2 is disabled in the current Security state, a write of 1 to this bit resets all the event counters.

- In EL2 and EL3, a write of 1 to this bit resets all the event counters.
- This field does not affect the operation of other event counters and PMCCNTR_EL0.

Note

Resetting the event counters does not change the event counter overflow bits. If FEAT_PMUv3p5 is implemented, the values of MDCR_EL2.HLP and PMCR_EL0.LP are ignored, and bits [63:0] of all affected event counters are reset.

Access to this field is **WO/RAZ**.

E, bit [0]

Enable.

In the description of this field:

- If EL2 is implemented, then PMN is MDCR_EL2.HPMN.
- If EL2 is not implemented, then PMN is PMCR_EL0.N.

E	Meaning
`0b0`	Affected counters are disabled and do not count.
`0b1`	Affected counters are enabled by PMCNTENSET_EL0.

The counters affected by this field are:

- If EL2 is implemented, event counters PMEVCNTR<n>_EL0 for values of n less than PMN. This applies even when EL2 is disabled in the current Security state.
- If EL2 is not implemented, all event counters PMEVCNTR<n>_EL0.
- If FEAT_PMUv3_ICNTR is implemented, the instruction counter PMICNTR_EL0.
- The cycle counter PMCCNTR_EL0.

Other event counters are not affected by this field.

The reset behavior of this field is:

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

Accessing PMCR_EL0

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

```
MRS <Xt>, PMCR_EL0
```

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCR_EL0;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCR_EL0;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
```

```
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = PMCR_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMCR_EL0;
```

MSR PMCR_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1001	0b1100	0b000

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif PMUSERENR_EL0.EN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11'
&& IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3)
|| SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCR_EL0 ==
'1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        PMCR_EL0 = X[t, 64];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && MDCR_EL3.TPM == '1' then
        UNDEFINED;
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.PMCR_EL0 == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPMCR == '1' then
```

```
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL2 then
        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
    && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
    when SDD == '1'" && MDCR_EL3.TPM == '1' then
            UNDEFINED;
        elsif HaveEL(EL3) && MDCR_EL3.TPM == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        else
            PMCR_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        PMCR_EL0 = X[t, 64];
```