PMEVTYPER<n>_EL0, Performance Monitors Event Type Registers, n = 0 - 30

The PMEVTYPER<n>_EL0 characteristics are:

Purpose

Configures event counter n, where n is 0 to 30.

Configuration

AArch64 System register PMEVTYPER<n>_EL0 bits [31:0] are architecturally mapped to AArch32 System register PMEVTYPER<n>[31:0].

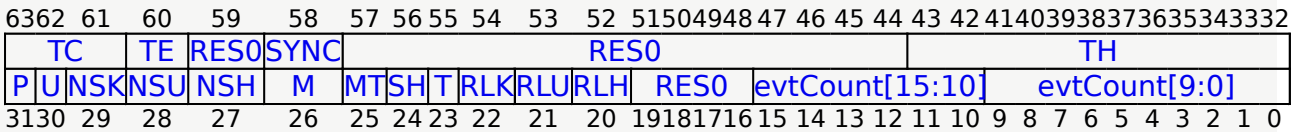
AArch64 System register PMEVTYPER<n>_EL0 bits [63:0] are architecturally mapped to External register PMU.PMEVTYPER<n>_EL0[63:0] when FEAT_PMUv3_TH is implemented, or FEAT_PMUv3p8 is implemented or FEAT_PMUv3_EXT64 is implemented.

This register is present only when FEAT_PMUv3 is implemented. Otherwise, direct accesses to PMEVTYPER<n>_EL0 are undefined.

Attributes

PMEVTYPER<n>_EL0 is a 64-bit register.

Field descriptions



TC, bits [63:61]
When (FEAT_PMUv3_EDGE is not implemented or PMEVTYPER<n>_EL0.TE == 0) and FEAT_PMUv3_TH is implemented:

Threshold Control.

Defines the threshold function. In the description of this field:

- V_B is the value the event specified by PMEVTYPER<n>_EL0 would increment the counter by on a processor cycle if the threshold function is disabled.

- TH is the value of PMEVTYPER<n>_EL0.TH.

Comparisons treat V_B and TH as unsigned integer values.

TC	Meaning
0b000	Not-equal. The counter increments by V_B on each processor cycle when V_B is not equal to TH. If TH is zero, the threshold function is disabled.
0b001	Not-equal, count. The counter increments by 1 on each processor cycle when V_B is not equal to TH.
0b010	Equals. The counter increments by V_B on each processor cycle when V_B is equal to TH.
0b011	Equals, count. The counter increments by 1 on each processor cycle when V_B is equal to TH.
0b100	Greater-than-or-equal. The counter increments by V_B on each processor cycle when V_B is greater than or equal to TH.
0b101	Greater-than-or-equal, count. The counter increments by 1 on each processor cycle when V_B is greater than or equal to TH.
0b110	Less-than. The counter increments by V_B on each processor cycle when V_B is less than TH.
0b111	Less-than, count. The counter increments by 1 on each processor cycle when V_B is less than TH.

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.

When FEAT_PMUv3_EDGE is implemented and PMEVTYPER<n>_EL0.TE == 1:

Threshold Control.

Defines the threshold function. In the description of this field:

- V_B is the value the event specified by PMEVTYPER<n>_EL0 would increment the counter by on a processor cycle if the threshold function is disabled.
- TH is the value of PMEVTYPER<n>_EL0.TH.

Comparisons treat V_B and TH as unsigned integer values.

TC	Meaning
0b001	Equal to not-equal. The counter increments by 1 on each processor cycle when V_B is not equal to TH and V_B was equal to TH on the previous processor cycle.
0b010	Equal to/from not-equal. The counter increments by 1 on each processor cycle when either: • V_B is not equal to TH and V_B was equal to TH on the previous processor cycle. • V_B is equal to TH and V_B was not equal to TH on the previous processor cycle.
0b011	Not-equal to equal. The counter increments by 1 on each processor cycle when V_B is equal to TH and V_B was not equal to TH on the previous processor cycle.
0b101	Less-than to greater-than-or-equal. The counter increments by 1 on each processor cycle when V_B is greater than or equal to TH and V_B was less than TH on the previous processor cycle.

`0b110`	Less-than to/from greater-than-or-equal. The counter increments by 1 on each processor cycle when either: • V_B is greater than or equal to TH and V_B was less than TH on the previous processor cycle. • V_B is less than TH and V_B was greater than or equal to TH on the previous processor cycle.
`0b111`	Greater-than-or-equal to less-than. The counter increments by 1 on each processor cycle when V_B is less than TH and V_B was greater than or equal to TH on the previous processor cycle.

All other values are reserved.

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.

Threshold Control.

Defines the threshold function. In the description of this field:

- V_B is the value the event specified by PMEVTYPER<n>_EL0 would increment the counter by on a processor cycle if the threshold function is disabled.
- TH is the value of PMEVTYPER<n>_EL0.TH.

Comparisons treat V_B and TH as unsigned integer values.

TE, bit [60]
When FEAT_PMUv3_EDGE is implemented:

Threshold Edge. Enables the edge condition. When PMEVTYPER<n>_EL0.TE is 1, the event counter increments on cycles when the result of the threshold condition changes. See PMEVTYPER<n>_EL0.TC for more information.

TE	Meaning
0b0	Threshold edge condition disabled.
0b1	Threshold edge condition enabled.

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.

Bit [59]

Reserved, res0.

SYNC, bit [58]
When FEAT_SEBEP is implemented:

Synchronous mode. Controls whether a PMU exception generated by the counter is synchronous or asynchronous.

SYNC	Meaning
0b0	Asynchronous PMU exception is enabled.
0b1	Synchronous PMU exception is enabled.

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.

Bits [57:44]

Reserved, res0.

TH, bits [43:32]
When FEAT_PMUv3_TH is implemented:

Threshold value. Provides the unsigned value for the threshold function defined by PMEVTYPER<n>_EL0.TC.

If PMEVTYPER<n>_EL0.TC is `0b000` and PMEVTYPER<n>_EL0.TH is zero, then the threshold function is disabled.

If PMMIR_EL1.THWIDTH is less than 12, then bits PMEVTYPER<n>_EL0.TH[11:PMMIR_EL1.THWIDTH] are res0. This accounts for the behavior when writing a value greater-than-or-equal-to $2^{(\text{PMMIR\_EL1.THWIDTH})}$.

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.

P, bit [31]

EL1 filtering. Controls counting events in EL1.

P	Meaning
`0b0`	This field has no effect on filtering of events.
`0b1`	Events in EL1 are not counted.

If Secure and Non-secure states are implemented, then counting events in Non-secure EL1 is further controlled by PMEVTYPER<n>_EL0.NSK.

If FEAT_RME is implemented, then counting events in Realm EL1 is further controlled by PMEVTYPER<n>_EL0.RLK.

If EL3 is implemented, then counting events in EL3 is further controlled by PMEVTYPER<n>_EL0.M.

The reset behavior of this field is:

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

U, bit [30]

EL0 filtering. Controls counting events in EL0.

U	Meaning
0b0	This field has no effect on filtering of events.
0b1	Events in EL0 are not counted.

If Secure and Non-secure states are implemented, then counting events in Non-secure EL0 is further controlled by PMEVTYPER<n>_EL0.NSU.

If FEAT_RME is implemented, then counting events in Realm EL0 is further controlled by PMEVTYPER<n>_EL0.RLU.

The reset behavior of this field is:

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

NSK, bit [29]
When EL3 is implemented:

Non-secure EL1 filtering. Controls counting events in Non-secure EL1. If PMEVTYPER<n>_EL0.NSK is not equal to PMEVTYPER<n>_EL0.P, then events in Non-secure EL1 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSK has no effect on filtering of events in Non-secure EL1.

NSK	Meaning
0b0	When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in Non-secure EL1 are not counted.
0b1	When PMEVTYPER<n>_EL0.P == 0, events in Non-secure EL1 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

NSU, bit [28]
When EL3 is implemented:

Non-secure EL0 filtering. Controls counting events in Non-secure EL0. If PMEVTYPER<n>_EL0.NSU is not equal to PMEVTYPER<n>_EL0.U, then events in Non-secure EL0 are not counted. Otherwise, PMEVTYPER<n>_EL0.NSU has no effect on filtering of events in Non-secure EL0.

NSU	Meaning
0b0	When PMEVTYPER<n>_EL0.U == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.U == 1, events in Non-secure EL0 are not counted.
0b1	When PMEVTYPER<n>_EL0.U == 0, events in Non-secure EL0 are not counted. When PMEVTYPER<n>_EL0.U == 1, this field has no effect on filtering of events.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

NSH, bit [27]
When EL2 is implemented:

EL2 filtering. Controls counting events in EL2.

NSH	Meaning
0b0	Events in EL2 are not counted.

0b1	This field has no effect on filtering of events.

If EL3 is implemented and FEAT_SEL2 is implemented, then counting events in Secure EL2 is further controlled by PMEVTYPER<n>_EL0.SH.

If FEAT_RME is implemented, then counting events in Realm EL2 is further controlled by PMEVTYPER<n>_EL0.RLH.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

M, bit [26]
When EL3 is implemented:

EL3 filtering. Controls counting events in EL3. If PMEVTYPER<n>_EL0.M is not equal to PMEVTYPER<n>_EL0.P, then events in EL3 are not counted. Otherwise, PMEVTYPER<n>_EL0.M has no effect on filtering of events in EL3.

M	Meaning
0b0	When PMEVTYPER<n>_EL0.P == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.P == 1, events in EL3 are not counted.
0b1	When PMEVTYPER<n>_EL0.P == 0, events in EL3 are not counted. When PMEVTYPER<n>_EL0.P == 1, this field has no effect on filtering of events.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

MT, bit [25]
When FEAT_MTPMU is implemented or an IMPLEMENTATION DEFINED multi-threaded PMU extension is implemented:

Multithreading.

MT	Meaning
0b0	Count events only on controlling PE.
0b1	Count events from any PE with the same affinity at level 1 and above as this PE.

From Armv8.6, the implementation defined multi-threaded PMU extension is not permitted, meaning if FEAT_MTPMU is not implemented, this field is res0. See ID_AA64DFR0_EL1.MTPMU.

This field is ignored by the PE and treated as zero when FEAT_MTPMU is implemented and Disabled.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

SH, bit [24]
When EL3 is implemented and FEAT_SEL2 is implemented:

Secure EL2 filtering. Controls counting events in Secure EL2. If PMEVTYPER<n>_EL0.SH is equal to PMEVTYPER<n>_EL0.NSH, then events in Secure EL2 are not counted. Otherwise, PMEVTYPER<n>_EL0.SH has no effect on filtering of events in Secure EL2.

SH	Meaning
0b0	When PMEVTYPER<n>_EL0.NSH == 0, events in Secure EL2 are not counted. When PMEVTYPER<n>_EL0.NSH == 1, this field has no effect on filtering of events.
0b1	When PMEVTYPER<n>_EL0.NSH == 0, this field has no effect on filtering of events. When PMEVTYPER<n>_EL0.NSH == 1, events in Secure EL2 are not counted.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

T, bit [23]
When FEAT_TME is implemented:

Transactional state filtering bit. Controls counting of Attributable events in Non-transactional state.

T	Meaning
0b0	This bit has no effect on the filtering of events.
0b1	Do not count Attributable events in Non-transactional state.

For each Unattributable event, it is implementation defined whether the filtering applies.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

RLK, bit [22]
When FEAT_RME is implemented:

Realm EL1 (kernel) filtering bit. Controls counting in Realm EL1.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.P bit, events in Realm EL1 are counted.

Otherwise, events in Realm EL1 are not counted.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

RLU, bit [21]
When FEAT_RME is implemented:

Realm EL0 (unprivileged) filtering bit. Controls counting in Realm EL0.

If the value of this bit is equal to the value of the PMEVTYPER<n>_EL0.U bit, events in Realm EL0 are counted.

Otherwise, events in Realm EL0 are not counted.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

RLH, bit [20]
When FEAT_RME is implemented:

Realm EL2 filtering bit. Controls counting in Realm EL2.

If the value of this bit is not equal to the value of the PMEVTYPER<n>_EL0.NSH bit, events in Realm EL2 are counted.

Otherwise, events in Realm EL2 are not counted.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

Bits [19:16]

Reserved, res0.

evtCount[15:10], bits [15:10]
When FEAT_PMUv3p1 is implemented:

Extension to evtCount[9:0]. For more information, see evtCount[9:0].

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

evtCount[9:0], bits [9:0]

Event to count.

The event number of the event that is counted by event counter PMEVCNTR<n>_EL0.

The ranges of event numbers allocated to each type of event are shown in 'Allocation of the PMU event number space'.

If FEAT_PMUv3p8 is implemented and PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.

Note

Arm recommends this behavior for all implementations of FEAT_PMUv3.

Otherwise, if PMEVTYPER<n>_EL0.evtCount is programmed to an event that is reserved or not supported by the PE, the behavior depends on the value written:

- For the range `0x0000` to `0x003F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.
- If FEAT_PMUv3p1 is implemented, for the range `0x4000` to `0x403F`, no events are counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is the value written to the field.

- For other values, it is unpredictable what event, if any, is counted and the value returned by a direct or external read of the PMEVTYPER<n>_EL0.evtCount field is unknown.

Note

unpredictable means the event must not expose privileged information.

The reset behavior of this field is:

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

Accessing PMEVTYPER<n>_EL0

PMEVTYPER<n>_EL0 can also be accessed by using PMXEVTYPER_EL0 with PMSELR_EL0.SEL set to n.

If FEAT_FGT is implemented and <n> is greater than or equal to the number of accessible event counters, then the behavior of permitted reads and writes of PMEVTYPER<n>_EL0 is as follows:

- If <n> is an unimplemented event counter, the access is undefined.
- Otherwise, the access is trapped to EL2.

If FEAT_FGT is not implemented and <n> is greater than or equal to the number of accessible event counters, then reads and writes of PMEVTYPER<n>_EL0 are constrained unpredictable, and the following behaviors are permitted:

- Accesses to the register are undefined.
- Accesses to the register behave as RAZ/WI.
- Accesses to the register execute as a `NOP`.
- Accesses to the register behave as if <n> is an unknown value less-than-or-equal-to the index of the highest accessible event counter.
- If EL2 is implemented and enabled in the current Security state, and <n> is less than the number of implemented event counters, accesses from EL1 or permitted accesses from EL0 are trapped to EL2.

PMEVTYPER<n>_EL0 reads-as-zero and ignores writes if all of the following are true:

- FEAT_PMUv3p9 is implemented.
- PSTATE.EL == EL0.
- PMUSERENR_EL0.UEN == 1.
- PMUACR_EL1.P<n> == 0.

PMEVTYPER<n>_EL0 ignores writes if all of the following are true:

- FEAT_PMUv3p9 is implemented.

- PSTATE.EL == EL0.
- PMUSERENR_EL0.{UEN,ER} == {1,1}.

Note

In EL0, an access is permitted if it is enabled by PMUSERENR_EL0.{UEN,EN}.

If EL2 is implemented and enabled in the current Security state, in EL1 and EL0, MDCR_EL2.HPMN identifies the number of accessible event counters. Otherwise, the number of accessible event counters is the number of implemented event counters. For more information, see MDCR_EL2.HPMN.

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

MRS <Xt>, PMEVTYPER<m>_EL0 ; Where m = 0-30

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b11:m[4:3]	m[2:0]

```
integer m = UInt(CRm<1:0>:op2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else

ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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') &&
HDFGRTR_EL2.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif EL2Enabled() && m >=
AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then

ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            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] = PMEVTYPER_EL0[m];
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') && HDFGRTR_EL2.PMEVTYPERn_EL0
== '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && m >=
AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then

ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            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] = PMEVTYPER_EL0[m];
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] = PMEVTYPER_EL0[m];
elsif PSTATE.EL == EL3 then
    X[t, 64] = PMEVTYPER_EL0[m];
```

MSR PMEVTYPER<m>_EL0, <Xt> ; Where m = 0-30

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b11:m[4:3]	m[2:0]

```
integer m = UInt(CRm<1:0>:op2<2:0>);

if m >= NUM_PMU_COUNTERS then
    if IsFeatureImplemented(FEAT_FGT) then
        UNDEFINED;
    else

ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
elsif 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.PMEVTYPERn_EL0 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && m >=
AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then

ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            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
        PMEVTYPER_EL0[m] = 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.PMEVTYPERn_EL0
== '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif EL2Enabled() && MDCR_EL2.TPM == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && m >=
AArch64.GetNumEventCountersAccessible() then
        if !IsFeatureImplemented(FEAT_FGT) then

ConstrainUnpredictableProcedure(Unpredictable_PMUEVENTCOUNTER);
        else
            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
        PMEVTYPER_EL0[m] = 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
        PMEVTYPER_EL0[m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    PMEVTYPER_EL0[m] = X[t, 64];
```

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94