CNTHP_TVAL_EL2, Counter-timer Physical Timer TimerValue register (EL2)

The CNTHP_TVAL_EL2 characteristics are:

Purpose

Holds the timer value for the EL2 physical timer.

Configuration

AArch64 System register CNTHP_TVAL_EL2 bits [31:0] are architecturally mapped to AArch32 System register CNTHP_TVAL[31:0].

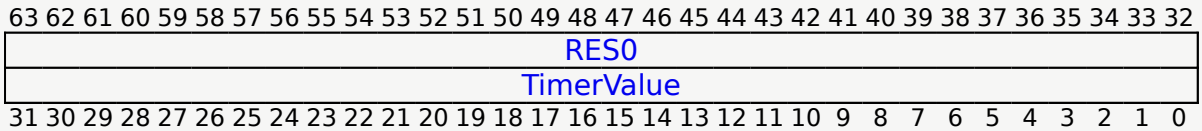
This register is present only when EL3 is implemented or (EL3 is not implemented, EL2 is implemented and FEAT_SEL2 is not implemented). Otherwise, direct accesses to CNTHP_TVAL_EL2 are undefined.

If EL2 is not implemented, this register is res0 from EL3.

Attributes

CNTHP_TVAL_EL2 is a 64-bit register.

Field descriptions



Bits [63:32]

Reserved, res0.

TimerValue, bits [31:0]

The TimerValue view of the EL2 physical timer.

On a read of this register:

- If CNTHP_CTL_EL2.ENABLE is 0, the value returned is unknown.
- If CNTHP_CTL_EL2.ENABLE is 1, the value returned is (CNTHP_CVAL_EL2 - CNTPCT_EL0).

On a write of this register, CNTHP_CVAL_EL2 is set to (CNTPCT_EL0 + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When CNTHP_CTL_EL2.ENABLE is 1, the timer condition is met when (CNTPCT_EL0 - CNTHP_CVAL_EL2) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- CNTHP_CTL_EL2.ISTATUS is set to 1.
- If CNTHP_CTL_EL2.IMASK is 0, an interrupt is generated.

When CNTHP_CTL_EL2.ENABLE is 0, the timer condition is not met, but CNTPCT_EL0 continues to count, so the TimerValue view appears to continue to count down.

The reset behavior of this field is:

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

Accessing CNTHP_TVAL_EL2

When HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHP_TVAL_EL2 or CNTP_TVAL_EL0 are not guaranteed to be ordered with respect to accesses using the other mnemonic.

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

MRS <Xt>, CNTHP_TVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if CNTHP_CTL_EL2.ENABLE == '0' then
        X[t, 64] = bits(64) UNKNOWN;
    else
        X[t, 64] = CNTHP_CVAL_EL2 -
PhysicalCountInt();
elsif PSTATE.EL == EL3 then
```

```
    if CNTHP_CTL_EL2.ENABLE == '0' then
        X[t, 64] = bits(64) UNKNOWN;
    else
        X[t, 64] = CNTHP_CVAL_EL2 -
PhysicalCountInt();
```

MSR CNTHP_TVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0010	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    CNTHP_CVAL_EL2 = SignExtend(X[t, 64]<31:0>, 64)
+ PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    CNTHP_CVAL_EL2 = SignExtend(X[t, 64]<31:0>, 64)
+ PhysicalCountInt();
```

MRS <Xt>, CNTP_TVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11')
&& CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' &&
CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10'
&& CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
```

```
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHPS_CVAL_EL2 -
PhysicalCountInt();
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '1' then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHP_CVAL_EL2 -
PhysicalCountInt();
    elsif IsFeatureImplemented(FEAT_ECV) &&
EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11'
then
        if CNTP_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTP_CVAL_EL0 -
(PhysicalCountInt() - CNTPOFF_EL2);
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTP_CVAL_EL0 -
PhysicalCountInt();
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.E2H == '0' &&
CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif IsFeatureImplemented(FEAT_ECV) &&
EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTP_CVAL_EL0 -
(PhysicalCountInt() - CNTPOFF_EL2);
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTP_CVAL_EL0 -
PhysicalCountInt();
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        if CNTHPS_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHPS_CVAL_EL2 -
PhysicalCountInt();
    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1'
```

```
then
        if CNTHP_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHP_CVAL_EL2 -
PhysicalCountInt();
    else
        if CNTP_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTP_CVAL_EL0 -
PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if CNTP_CTL_EL0.ENABLE == '0' then
        X[t, 64] = bits(64) UNKNOWN;
    else
        X[t, 64] = CNTP_CVAL_EL0 -
PhysicalCountInt();
```

MSR CNTP_TVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b000

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11')
&& CNTKCTL_EL1.EL0PTEN == '0' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x18);
        else
            AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && HCR_EL2.E2H == '0' &&
CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '10'
&& CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& CNTHCTL_EL2.EL0PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '1' then
        CNTHP_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
    elsif IsFeatureImplemented(FEAT_ECV) &&
EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' && HCR_EL2.<E2H,TGE> != '11'
then
```

```
        CNTP_CVAL_EL0 = (SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt()) - CNTPOFF_EL2;
    else
        CNTP_CVAL_EL0 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.E2H == '0' &&
CNTHCTL_EL2.EL1PCEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CNTHCTL_EL2.EL1PTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif IsFeatureImplemented(FEAT_ECV) &&
EL2Enabled() && SCR_EL3.ECVEn == '1' &&
CNTHCTL_EL2.ECV == '1' then
        CNTP_CVAL_EL0 = (SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt()) - CNTPOFF_EL2;
    else
        CNTP_CVAL_EL0 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1'
then
        CNTHP_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
    else
        CNTP_CVAL_EL0 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    CNTP_CVAL_EL0 = SignExtend(X[t, 64]<31:0>, 64) +
PhysicalCountInt();
```

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94