CNTP_TVAL_EL0, Counter-timer Physical Timer TimerValue Register

The CNTP_TVAL_EL0 characteristics are:

Purpose

Holds the timer value for the EL1 physical timer.

Configuration

AArch64 System register CNTP_TVAL_EL0 bits [31:0] are architecturally mapped to AArch32 System register CNTP_TVAL[31:0].

Attributes

CNTP_TVAL_EL0 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
RES0
TimerValue
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 [63:32]

Reserved, res0.

TimerValue, bits [31:0]

The TimerValue view of the EL1 physical timer.

On a read of this register:

- If CNTP_CTL_EL0.ENABLE is 0, the value returned is unknown.
- If CNTP_CTL_EL0.ENABLE is 1, the value returned is (CNTP_CVAL_EL0 - CNTPCT_EL0).

On a write of this register, CNTP_CVAL_EL0 is set to (CNTPCT_EL0 + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When CNTP_CTL_EL0.ENABLE is 1, the timer condition is met when (CNTPCT_EL0 - CNTP_CVAL_EL0) is greater than or equal to zero. This means that TimerValue acts like a 32-bit downcounter timer. When the timer condition is met:

- CNTP_CTL_EL0.ISTATUS is set to 1.
- If CNTP_CTL_EL0.IMASK is 0, an interrupt is generated.

When CNTP_CTL_EL0.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.

Note

The value of CNTPCT_EL0 used in these calculations is the value seen at the Exception Level that the CNTPCT_EL0 regsiter is being read or written from.

The reset behavior of this field is:

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

Accessing CNTP_TVAL_EL0

When HCR_EL2.E2H is 1, without explicit synchronization, access from EL3 using the mnemonic CNTP_TVAL_EL0 or CNTP_TVAL_EL02 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>, 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();
```

MRS <Xt>, CNTP_TVAL_EL02

op0	op1	CRn	CRm	op2
0b11	0b101	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 HCR_EL2.E2H == '1' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTP_CVAL_EL0 -
PhysicalCountInt();
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) &&
HCR_EL2.E2H == '1' then
        if CNTP_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTP_CVAL_EL0 -
PhysicalCountInt();
    else
        UNDEFINED;
```

MSR CNTP_TVAL_EL02, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b101	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 HCR_EL2.E2H == '1' then
        CNTP_CVAL_EL0 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
    else
        UNDEFINED;
elsif PSTATE.EL == EL3 then
    if EL2Enabled() && !ELUsingAArch32(EL2) &&
HCR_EL2.E2H == '1' then
        CNTP_CVAL_EL0 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
    else
        UNDEFINED;
```

28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94