CNTHVS_TVAL_EL2, Counter-timer Secure Virtual Timer TimerValue register (EL2)

The CNTHVS_TVAL_EL2 characteristics are:

Purpose

Holds the timer value for the Secure EL2 virtual timer.

Configuration

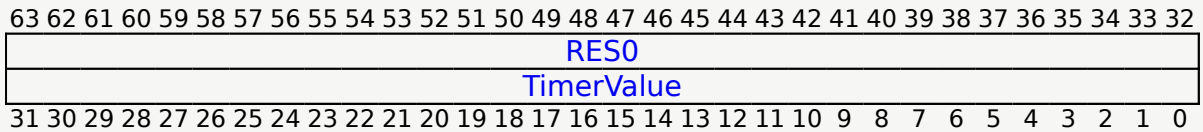
AArch64 System register CNTHVS_TVAL_EL2 bits [31:0] are architecturally mapped to AArch32 System register CNTHVS_TVAL[31:0].

This register is present only when FEAT_SEL2 is implemented and FEAT_VHE is implemented. Otherwise, direct accesses to CNTHVS_TVAL_EL2 are undefined.

Attributes

CNTHVS_TVAL_EL2 is a 64-bit register.

Field descriptions



Bits [63:32]

Reserved, res0.

TimerValue, bits [31:0]

The TimerValue view of the EL2 virtual timer.

On a read of this register:

- If CNTHVS_CTL_EL2.ENABLE is 0, the value returned is unknown.
- If CNTHVS_CTL_EL2.ENABLE is 1, the value returned is (CNTHVS_CVAL_EL2 - CNTVCT_EL0).

On a write of this register, CNTHVS_CVAL_EL2 is set to (CNTVCT_EL0 + TimerValue), where TimerValue is treated as a signed 32-bit integer.

When CNTHVS_CTL_EL2.ENABLE is 1, the timer condition is met when ((CNTVCT_EL0 - CNTHVS_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:

- CNTHVS_CTL_EL2.ISTATUS is set to 1.
- If CNTHVS_CTL_EL2.IMASK is 0, an interrupt is generated.

When CNTHVS_CTL_EL2.ENABLE is 0, the timer condition is not met, but CNTVCT_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 CNTHVS_TVAL_EL2

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

MRS <Xt>, CNTHVS_TVAL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0100	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    else
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHVS_CVAL_EL2 -
PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
```

```
        UNDEFINED;
    else
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHVS_CVAL_EL2 -
PhysicalCountInt();
```

MSR CNTHVS_TVAL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0100	0b000

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.NV == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        UNDEFINED;
elsif PSTATE.EL == EL2 then
    if !IsCurrentSecurityState(SS_Secure) then
        UNDEFINED;
    else
        CNTHVS_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if SCR_EL3.EEL2 == '0' then
        UNDEFINED;
    else
        CNTHVS_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
```

MRS <Xt>, CNTV_TVAL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11')
&& CNTKCTL_EL1.EL0VTEN == '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'
&& CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11'
&& CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHVS_CVAL_EL2 -
PhysicalCountInt();
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '1' then
        if CNTHV_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHV_CVAL_EL2 -
PhysicalCountInt();
    elsif HaveEL(EL2) && (!EL2Enabled() ||
HCR_EL2.<E2H,TGE> != '11') then
        if CNTV_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTV_CVAL_EL0 -
(PhysicalCountInt() - CNTVOFF_EL2);
    else
        if CNTV_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTV_CVAL_EL0 -
PhysicalCountInt();
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL2) then
        if CNTV_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTV_CVAL_EL0 -
(PhysicalCountInt() - CNTVOFF_EL2);
    else
        if CNTV_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTV_CVAL_EL0 -
PhysicalCountInt();
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        if CNTHVS_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHVS_CVAL_EL2 -
PhysicalCountInt();
    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1'
then
```

```
        if CNTHV_CTL_EL2.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTHV_CVAL_EL2 -
PhysicalCountInt();
    elsif HCR_EL2.E2H == '0' then
        if CNTV_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTV_CVAL_EL0 -
(PhysicalCountInt() - CNTVOFF_EL2);
    else
        if CNTV_CTL_EL0.ENABLE == '0' then
            X[t, 64] = bits(64) UNKNOWN;
        else
            X[t, 64] = CNTV_CVAL_EL0 -
PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if CNTV_CTL_EL0.ENABLE == '0' then
        X[t, 64] = bits(64) UNKNOWN;
    elsif HaveEL(EL2) && !ELUsingAArch32(EL2) then
        X[t, 64] = CNTV_CVAL_EL0 -
(PhysicalCountInt() - CNTVOFF_EL2);
    elsif HaveEL(EL2) && ELUsingAArch32(EL2) then
        X[t, 64] = CNTV_CVAL_EL0 -
(PhysicalCountInt() - CNTVOFF);
    else
        X[t, 64] = CNTV_CVAL_EL0 -
PhysicalCountInt();
```

MSR CNTV_TVAL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0011	0b000

```
if PSTATE.EL == EL0 then
    if !(EL2Enabled() && HCR_EL2.<E2H,TGE> == '11')
&& CNTKCTL_EL1.EL0VTEN == '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'
&& CNTHCTL_EL2.EL0VTEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> != '11'
&& CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHVS_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
```

```
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '1' then
        CNTHV_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
    elsif HaveEL(EL2) && (!EL2Enabled() ||
HCR_EL2.<E2H,TGE> != '11') then
        CNTV_CVAL_EL0 = (SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt()) - CNTVOFF_EL2;
    else
        CNTV_CVAL_EL0 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && CNTHCTL_EL2.EL1TVT == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL2) then
        CNTV_CVAL_EL0 = (SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt()) - CNTVOFF_EL2;
    else
        CNTV_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
        CNTHVS_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1'
then
        CNTHV_CVAL_EL2 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
    elsif HCR_EL2.E2H == '0' then
        CNTV_CVAL_EL0 = (SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt()) - CNTVOFF_EL2;
    else
        CNTV_CVAL_EL0 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
elsif PSTATE.EL == EL3 then
    if HaveEL(EL2) && !ELUsingAArch32(EL2) then
        CNTV_CVAL_EL0 = (SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt()) - CNTVOFF_EL2;
    elsif HaveEL(EL2) && ELUsingAArch32(EL2) then
        CNTV_CVAL_EL0 = (SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt()) - CNTVOFF;
    else
        CNTV_CVAL_EL0 = SignExtend(X[t, 64]<31:0>,
64) + PhysicalCountInt();
```


28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94