CNTHP_CTL_EL2, Counter-timer Hypervisor Physical Timer Control register

The CNTHP_CTL_EL2 characteristics are:

Purpose

Control register for the EL2 physical timer.

Configuration

AArch64 System register CNTHP_CTL_EL2 bits [31:0] are architecturally mapped to AArch32 System register CNTHP_CTL[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_CTL_EL2 are undefined.

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

Attributes

CNTHP_CTL_EL2 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																															
RES0																													ISTATUS	IMASK	ENABLE

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 [63:3]

Reserved, res0.

ISTATUS, bit [2]

The status of the timer. This bit indicates whether the timer condition is met:

ISTATUS	Meaning
0b0	Timer condition is not met.
0b1	Timer condition is met.

When the value of the ENABLE bit is 1, ISTATUS indicates whether the timer condition is met. ISTATUS takes no account of the value of the IMASK bit. If the value of ISTATUS is 1 and the value of IMASK is 0 then the timer interrupt is asserted.

When the value of the ENABLE bit is 0, the ISTATUS field is unknown.

The reset behavior of this field is:

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

Access to this field is **RO**.

IMASK, bit [1]

Timer interrupt mask bit. Permitted values are:

IMASK	Meaning
`0b0`	Timer interrupt is not masked by the IMASK bit.
`0b1`	Timer interrupt is masked by the IMASK bit.

For more information, see the description of the ISTATUS bit.

The reset behavior of this field is:

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

ENABLE, bit [0]

Enables the timer. Permitted values are:

ENABLE	Meaning
`0b0`	Timer disabled.
`0b1`	Timer enabled.

Setting this bit to 0 disables the timer output signal, but the timer value accessible from CNTHP_TVAL_EL2 continues to count down.

Note

Disabling the output signal might be a power-saving option.

The reset behavior of this field is:

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

Accessing CNTHP_CTL_EL2

When HCR_EL2.E2H is 1, without explicit synchronization, access from EL2 using the mnemonic CNTHP_CTL_EL2 or CNTP_CTL_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_CTL_EL2

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0010	0b001

```
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
    X[t, 64] = CNTHP_CTL_EL2;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CNTHP_CTL_EL2;
```

MSR CNTHP_CTL_EL2, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b100	0b1110	0b0010	0b001

```
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_CTL_EL2 = X[t, 64];
elsif PSTATE.EL == EL3 then
    CNTHP_CTL_EL2 = X[t, 64];
```

MRS <Xt>, CNTP_CTL_EL0

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b001

```
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
        X[t, 64] = CNTHPS_CTL_EL2;
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '1' then
        X[t, 64] = CNTHP_CTL_EL2;
    else
        X[t, 64] = CNTP_CTL_EL0;
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 EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        X[t, 64] = NVMem[0x180];
    else
        X[t, 64] = CNTP_CTL_EL0;
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        X[t, 64] = CNTHPS_CTL_EL2;
    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1'
then
        X[t, 64] = CNTHP_CTL_EL2;
```

```
    else
        X[t, 64] = CNTP_CTL_EL0;
elsif PSTATE.EL == EL3 then
    X[t, 64] = CNTP_CTL_EL0;
```

MSR CNTP_CTL_EL0, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b1110	0b0010	0b001

```
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_CTL_EL2 = X[t, 64];
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& SCR_EL3.NS == '1' then
        CNTHP_CTL_EL2 = X[t, 64];
    else
        CNTP_CTL_EL0 = X[t, 64];
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 EL2Enabled() && HCR_EL2.<NV2,NV1,NV> ==
'111' then
        NVMem[0x180] = X[t, 64];
    else
        CNTP_CTL_EL0 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if HCR_EL2.E2H == '1' && SCR_EL3.NS == '0' &&
IsFeatureImplemented(FEAT_SEL2) then
        CNTHPS_CTL_EL2 = X[t, 64];
    elsif HCR_EL2.E2H == '1' && SCR_EL3.NS == '1'
then
```

```
            CNTHP_CTL_EL2 = X[t, 64];
        else
            CNTP_CTL_EL0 = X[t, 64];
    elsif PSTATE.EL == EL3 then
        CNTP_CTL_EL0 = X[t, 64];
```

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94