TRCVIPCSSCTLR, ViewInst Start/Stop PE Comparator Control Register

The TRCVIPCSSCTLR characteristics are:

Purpose

Use this to select, or read, which PE Comparator Inputs can control the ViewInst start/stop function.

Configuration

AArch64 System register TRCVIPCSSCTLR bits [31:0] are architecturally mapped to External register TRCVIPCSSCTLR[31:0].

This register is present only when FEAT_ETE is implemented, FEAT_TRC_SR is implemented and UInt(TRCIDR4.NUMPC) > 0. Otherwise, direct accesses to TRCVIPCSSCTLR are undefined.

Attributes

TRCVIPCSSCTLR 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
RES0																							

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8
RES0								STOP[7]	STOP[6]	STOP[5]	STOP[4]	STOP[3]	STOP[2]	STOP[1]	STOP[0]	RES0							

Bits [63:24]

Reserved, res0.

STOP[<m>], bit [m+16], for m = 7 to 0

Selects whether PE Comparator Input <m> is in use with ViewInst start/stop function, for the purpose of stopping trace.

STOP[<m>]	Meaning
0b0	The PE Comparator Input <m>, is not selected as a stop resource.

0b1	The PE Comparator Input <m>, is selected as a stop resource.

This bit is res0 if m >= TRCIDR4.NUMPC.

The reset behavior of this field is:

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

Bits [15:8]

Reserved, res0.

START[<m>], bit [m], for m = 7 to 0

Selects whether PE Comparator Input <m> is in use with ViewInst start/stop function, for the purpose of starting trace.

START[<m>]	Meaning
0b0	The PE Comparator Input <m>, is not selected as a start resource.
0b1	The PE Comparator Input <m>, is selected as a start resource.

This bit is res0 if m >= TRCIDR4.NUMPC.

The reset behavior of this field is:

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

Accessing TRCVIPCSSCTLR

Must be programmed if TRCIDR4.NUMPC != 0b0000.

Writes are constrained unpredictable if the trace unit is not in the Idle state.

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

```
MRS <Xt>, TRCVIPCSSCTLR
```

op0	op1	CRn	CRm	op2

0b10	0b001	0b0000	0b0011	0b010

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVIPCSSCTLR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVIPCSSCTLR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = TRCVIPCSSCTLR;
```

MSR TRCVIPCSSCTLR, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b001	0b0000	0b0011	0b010

```
if PSTATE.EL == EL0 then
    UNDEFINED;
elsif PSTATE.EL == EL1 then
```

```
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPACR_EL1.TTA == '1' then
        AArch64.SystemAccessTrap(EL1, 0x18);
    elsif EL2Enabled() && CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.TRC == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVIPCSSCTLR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TTA == '1' then
        UNDEFINED;
    elsif CPTR_EL2.TTA == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && CPTR_EL3.TTA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVIPCSSCTLR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TTA == '1' then
        AArch64.SystemAccessTrap(EL3, 0x18);
    else
        TRCVIPCSSCTLR = X[t, 64];
```


28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94