MDSELR_EL1, Breakpoint and Watchpoint Selection Register

The MDSELR_EL1 characteristics are:

Purpose

Selects the current breakpoints or watchpoints accessed by System register instructions.

Configuration

This register is present only when FEAT_Debugv8p9 is implemented. Otherwise, direct accesses to MDSELR_EL1 are undefined.

Attributes

MDSELR_EL1 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

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
RES0	BANK	RES0

Bits [63:6]

Reserved, res0.

BANK, bits [5:4]

Breakpoint and watchpoint bank select.

BANK	Meaning	Applies when
`0b00`	Select 0 to 15.	
`0b01`	Select 16 to 31.	When NUM_BREAKPOINTS > 16 or NUM_WATCHPOINTS > 16

`0b10`	Select 32 to 47.	When NUM_BREAKPOINTS > 32 or NUM_WATCHPOINTS > 32
`0b11`	Select 48 to 63.	When NUM_BREAKPOINTS > 48 or NUM_WATCHPOINTS > 48

Each of the following register names accesses a register for breakpoint or watchpoint <n>, where n = UInt(MDSELR_EL1.BANK:m[3:0]):

- DBGBCR<m>_EL1.
- DBGBVR<m>_EL1.
- DBGWCR<m>_EL1.
- DBGWVR<m>_EL1.

This field is ignored by the PE and treated as zeros when the Effective value of MDSCR_EL1.EBWE is 0.

The reset behavior of this field is:

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

Accessing this field has the following behavior:

- Access is **RES0** if all of the following are true:
    - NUM_BREAKPOINTS <= 16
    - NUM_WATCHPOINTS <= 16
- Otherwise, access to this field is **RW**.

Bits [3:0]

Reserved, res0.

Accessing MDSELR_EL1

When 16 or fewer breakpoints are implemented, 16 or fewer watchpoints are implemented, and MDSELR_EL1 is implemented as RAZ/WI, it is implementation defined whether these trap controls have any effect on accesses to MDSELR_EL1.

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

MRS <Xt>, MDSELR_EL1

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0100	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'" && MDCR_EL3.EBWE == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD ==
'1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT2) && HaveEL(EL3) &&
SCR_EL3.FGTEn2 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT2) &&
HDFGRTR2_EL2.nMDSELR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.EBWE == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MDSELR_EL1;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && MDCR_EL3.EBWE == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD ==
'1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.EBWE == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        X[t, 64] = MDSELR_EL1;
elsif PSTATE.EL == EL3 then
    X[t, 64] = MDSELR_EL1;
```

MSR MDSELR_EL1, <Xt>

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	0b0100	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'" && MDCR_EL3.EBWE == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD ==
'1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT2) && HaveEL(EL3) &&
SCR_EL3.FGTEn2 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT2) &&
HDFGWTR2_EL2.nMDSELR_EL1 == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.EBWE == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MDSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && MDCR_EL3.EBWE == '0' then
        UNDEFINED;
    elsif Halted() && HaveEL(EL3) && EDSCR.SDD ==
'1' && boolean IMPLEMENTATION_DEFINED "EL3 trap
```

```
priority when SDD == '1'" && MDCR_EL3.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.EBWE == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    else
        MDSELR_EL1 = X[t, 64];
elsif PSTATE.EL == EL3 then
    MDSELR_EL1 = X[t, 64];
```

AArch32 Registers   AArch64 Registers   AArch32 Instructions   AArch64 Instructions   Index by Encoding   External Registers