# DBGBCR<n>_EL1, Debug Breakpoint Control Registers, n = 0 - 63

The DBGBCR<n>_EL1 characteristics are:

## Purpose

Holds control information for a breakpoint. Forms breakpoint n together with value register DBGBVR<n>_EL1.

## Configuration

AArch64 System register DBGBCR<n>_EL1 bits [31:0] are architecturally mapped to AArch32 System register DBGBCR<n>[31:0].

AArch64 System register DBGBCR<n>_EL1 bits [63:0] are architecturally mapped to External register DBGBCR<n>_EL1[63:0].

If breakpoint n is not implemented, accesses to this register are undefined.

## Attributes

DBGBCR<n>_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 | | | | | | | | | | | | |
| LBNX | SSCE | MASK | BT | LBN | SSC | HMC | RES0 | BAS | RES0 | BT2 | PMC | E |
| 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.

**LBNX, bits [31:30]**
**When FEAT_Debugv8p9 is implemented:**

Linked Breakpoint Number.

For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBN, specifies the index of the breakpoint linked to.

For all other breakpoint types, this field is ignored and reads of the register return an unknown value.

This field extends DBGBCR<n>_EL1.LBN to support up to 64 implemented breakpoints.

The reset behavior of this field is:

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

**Otherwise:**

Reserved, res0.

**SSCE, bit [29]**
**When FEAT_RME is implemented:**

Security State Control Extended.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

The reset behavior of this field is:

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

**Otherwise:**

Reserved, res0.

**MASK, bits [28:24]**
**When FEAT_ABLE is implemented:**

Address Mask. Only address ranges up to 2GB can be watched using a single mask.

| MASK | Meaning |
|---|---|
| `0b00000` | No mask. |
| `0b00011..0b11111` | Number of address bits masked. |

All other values are reserved.

Indicates the number of masked address bits, from `0b00011` masking 3 address bits (`0x00000007` mask for address) to `0b11111` masking 31 address bits (`0x7FFFFFFF` mask for address).

If programmed with a reserved value, the breakpoint behaves as if either:

- DBGBCR<n>_EL1.MASK has been programmed with a defined value, which might be 0 (no mask), other than for a direct read of DBGBCR<n>_EL1.
- The breakpoint is disabled.

The reset behavior of this field is:

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

**Otherwise:**

Reserved, res0.

**BT, bits [23:20]**

Breakpoint Type.

With DBGBCR<n>_EL1.BT2 when implemented, specifies breakpoint type.

| BT | Meaning | Applies when |
|---|---|---|
| `0b0000` | Unlinked instruction address match. DBGBVR<n>_EL1 is the address of an instruction. | |
| `0b0001` | Linked instruction address match. As `0b0000`, but linked to a breakpoint that has linking enabled. | |
| `0b0010` | Unlinked Context ID match. When FEAT_VHE is implemented, EL2 is using AArch64, the Effective value of HCR_EL2.E2H is 1, and either the PE is executing at EL0 with HCR_EL2.TGE set to 1 or the PE is executing at EL2, then DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL2 value. Otherwise, DBGBVR<n>_EL1.ContextID must match the CONTEXTIDR_EL1 value. | When breakpoint n is context-aware |
| `0b0011` | As `0b0010`, with linking enabled. | When breakpoint n is context-aware |

| | | |
|---|---|---|
| `0b0100` | Unlinked instruction address mismatch. DBGBVR<n>_EL1 is the address of an instruction. | When FEAT_ABLE is implemented |
| `0b0101` | Linked instruction address mismatch. As `0b0100`, but linked to a breakpoint that has linking enabled. | When FEAT_ABLE is implemented |
| `0b0110` | Unlinked CONTEXTIDR_EL1 match. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1. | When FEAT_VHE is implemented and breakpoint n is context-aware |
| `0b0111` | As `0b0110`, with linking enabled. | When FEAT_VHE is implemented and breakpoint n is context-aware |
| `0b1000` | Unlinked VMID match. DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID. | When EL2 is implemented and breakpoint n is context-aware |
| `0b1001` | As `0b1000`, with linking enabled. | When EL2 is implemented and breakpoint n is context-aware |
| `0b1010` | Unlinked VMID and Context ID match. DBGBVR<n>_EL1.ContextID is a Context ID compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.VMID is a VMID compared against VTTBR_EL2.VMID. | When EL2 is implemented and breakpoint n is context-aware |
| `0b1011` | As `0b1010`, with linking enabled. | When EL2 is implemented and breakpoint n is context-aware |

| | | |
|---|---|---|
| `0b1100` | Unlinked CONTEXTIDR_EL2 match. DBGBVR<n>_EL1.ContextID2 is a Context ID compared against CONTEXTIDR_EL2. | When FEAT_VHE is implemented and breakpoint n is context-aware |
| `0b1101` | As `0b1100`, with linking enabled. | When FEAT_VHE is implemented and breakpoint n is context-aware |
| `0b1110` | Unlinked Full Context ID match. DBGBVR<n>_EL1.ContextID is compared against CONTEXTIDR_EL1, and DBGBVR<n>_EL1.ContextID2 is compared against CONTEXTIDR_EL2. | When FEAT_VHE is implemented and breakpoint n is context-aware |
| `0b1111` | As `0b1110`, with linking enabled. | When FEAT_VHE is implemented and breakpoint n is context-aware |

The reset behavior of this field is:

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

**LBN, bits [19:16]**

Linked Breakpoint Number.

For Linked address matching breakpoints, with DBGBCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to.

For all other breakpoint types, this field is ignored and reads of the register return an unknown value.

The reset behavior of this field is:

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

**SSC, bits [15:14]**

Security state control. Determines the Security states under which a Breakpoint debug event for breakpoint n is generated.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information on the effect of programming the fields to a reserved set of values, see 'Reserved DBGBCR<n>_EL1.{SSC, HMC, PMC} values'.

The reset behavior of this field is:

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

**HMC, bit [13]**

Higher mode control. Determines the debug perspective for deciding when a Breakpoint debug event for breakpoint n is generated.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information, see DBGBCR<n>_EL1.SSC.

The reset behavior of this field is:

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

**Bits [12:9]**

Reserved, res0.

**BAS, bits [8:5]**
**When AArch32 is supported:**

Byte address select. Defines which half-words an address-matching breakpoint matches, regardless of the instruction set and Execution state.

The permitted values depend on the breakpoint type.

For Address match breakpoints, the permitted values are:

| BAS | Match instruction at | Constraint for debuggers |
|---|---|---|
| `0b0011` | DBGBVR<n>_EL1 | Use for T32 instructions |
| `0b1100` | DBGBVR<n>_EL1 + 2 | Use for T32 instructions |
| `0b1111` | DBGBVR<n>_EL1 | Use for A64 and A32 instructions |

All other values are reserved. For more information, see 'Reserved DBGBCR<n>_EL1.BAS values'.

For more information on using the BAS field in address match breakpoints, see 'Using the BAS field in Address Match breakpoints'.

For Context matching breakpoints, this field is res1 and ignored.

The reset behavior of this field is:

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

**Otherwise:**

Reserved, res1.

**Bit [4]**

Reserved, res0.

**BT2, bit [3]**
**When FEAT_ABLE is implemented:**

Breakpoint Type 2. With DBGBCR<n>_EL1.BT, specifies breakpoint type.

| BT2 | Meaning |
|---|---|
| `0b0` | As DBGBCR<n>_EL1.BT. |

| | |
|---|---|
| `0b1` | As DBGBCR<n>_EL1.BT, but with linking enabled.<br>This value is only defined for the following DBGBCR<n>_EL1.BT values:<br>`0b0000`, `0b0001`, `0b0100`, and `0b0101`.<br>All other values are reserved. |

The reset behavior of this field is:

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

**Otherwise:**

Reserved, res0.

**PMC, bits [2:1]**

Privilege mode control. Determines the Exception level or levels at which a Breakpoint debug event for breakpoint n is generated.

The fields that indicate when the breakpoint can be generated are: HMC, PMC, SSC, and SSCE. These fields must be considered in combination, and the values that are permitted for these fields are constrained.

For more information on the operation of these fields, see 'Execution conditions for which a breakpoint generates Breakpoint exceptions'.

For more information, see DBGBCR<n>_EL1.SSC.

The reset behavior of this field is:

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

**E, bit [0]**

Enable breakpoint n.

| E | Meaning |
|---|---|
| `0b0` | Breakpoint n disabled. |
| `0b1` | Breakpoint n enabled. |

This field is ignored by the PE and treated as zero when all of the following are true:

- Any of the following are true:
    - `HaltOnBreakpointOrWatchpoint()` is FALSE and the Effective value of MDSCR_EL1.EBWE is 0.
    - `HaltOnBreakpointOrWatchpoint()` is TRUE and the Effective value of EDSCR2.EBWE is 0.
- FEAT_Debugv8p9 is implemented.
- n >= 16.

The reset behavior of this field is:

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

## Accessing DBGBCR<n>_EL1

When FEAT_Debugv8p9 is implemented, a PE is permitted to support up to 64 implemented breakpoints.

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

```
MRS <Xt>, DBGBCR<m>_EL1 ; Where m = 0-15
```

| op0 | op1 | CRn | CRm | op2 |
|---|---|---|---|---|
| 0b10 | 0b000 | 0b0000 | m[3:0] | 0b101 |

```
integer m = UInt(CRm<3:0>);

if (!IsFeatureImplemented(FEAT_Debugv8p9) && m >=
NUM_BREAKPOINTS) ||
(IsFeatureImplemented(FEAT_Debugv8p9) && m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16) >=
NUM_BREAKPOINTS) then
    UNDEFINED;
elsif 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.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HDFGRTR_EL2.DBGBCRn_EL1 ==
'1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00'
then
```

```
            AArch64.SystemAccessTrap(EL2, 0x18);
        elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
        elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed()
&& EDSCR.TDA == '1' then
            Halt(DebugHalt_SoftwareAccess);
        else
            if IsFeatureImplemented(FEAT_Debugv8p9) then
                X[t, 64] = DBGBCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)];
            else
                X[t, 64] = DBGBCR_EL1[m];
elsif PSTATE.EL == EL2 then
    if 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.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed()
&& EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        if IsFeatureImplemented(FEAT_Debugv8p9) then
            X[t, 64] = DBGBCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)];
        else
            X[t, 64] = DBGBCR_EL1[m];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() &&
EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        if IsFeatureImplemented(FEAT_Debugv8p9) then
            X[t, 64] = DBGBCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)];
        else
            X[t, 64] = DBGBCR_EL1[m];
```

## MSR DBGBCR<m>_EL1, <Xt> ; Where m = 0-15

| op0 | op1 | CRn | CRm | op2 |
|---|---|---|---|---|
| 0b10 | 0b000 | 0b0000 | m[3:0] | 0b101 |

```
integer m = UInt(CRm<3:0>);

if (!IsFeatureImplemented(FEAT_Debugv8p9) && m >=
NUM_BREAKPOINTS) ||
(IsFeatureImplemented(FEAT_Debugv8p9) && m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16) >=
NUM_BREAKPOINTS) then
    UNDEFINED;
elsif 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.TDA == '1' then
        UNDEFINED;
    elsif EL2Enabled() &&
IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
SCR_EL3.FGTEn == '1') && HDFGWTR_EL2.DBGBCRn_EL1 ==
'1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() && MDCR_EL2.<TDE,TDA> != '00'
then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed()
&& EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        if IsFeatureImplemented(FEAT_Debugv8p9) then
            DBGBCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)] = X[t, 64];
        else
            DBGBCR_EL1[m] = 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.TDA == '1' then
        UNDEFINED;
    elsif HaveEL(EL3) && MDCR_EL3.TDA == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
    elsif OSLSR_EL1.OSLK == '0' && HaltingAllowed()
&& EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        if IsFeatureImplemented(FEAT_Debugv8p9) then
            DBGBCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)] = X[t, 64];
        else
            DBGBCR_EL1[m] = X[t, 64];
elsif PSTATE.EL == EL3 then
    if OSLSR_EL1.OSLK == '0' && HaltingAllowed() &&
EDSCR.TDA == '1' then
        Halt(DebugHalt_SoftwareAccess);
    else
        if IsFeatureImplemented(FEAT_Debugv8p9) then
            DBGBCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)] = X[t, 64];
        else
            DBGBCR_EL1[m] = X[t, 64];
```

28/03/2023 16:01; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94