DBGBVR<n>_EL1, Debug Breakpoint Value Registers, n = 0 - 63

The DBGBVR<n>_EL1 characteristics are:

Purpose

Holds a virtual address, or a VMID and/or a context ID, for use in breakpoint matching. Forms breakpoint n together with control register DBGBCR<n>_EL1.

Configuration

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

AArch64 System register DBGBVR<n>_EL1 bits [63:32] are architecturally mapped to AArch32 System register DBGBXVR<n>[31:0].

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

How this register is interpreted depends on the value of DBGBCR<n>_EL1.BT.

- When DBGBCR<n>_EL1.BT is `0b000x`, this register holds a virtual address.
- When DBGBCR<n>_EL1.BT is `0b001x`, `0b011x`, or `0b110x`, this register holds a Context ID.
- When DBGBCR<n>_EL1.BT is `0b100x`, this register holds a VMID.
- When DBGBCR<n>_EL1.BT is `0b101x`, this register holds a VMID and a Context ID.
- When DBGBCR<n>_EL1.BT is `0b111x`, this register holds two Context ID values.

For other values of DBGBCR<n>_EL1.BT, this register is res0.

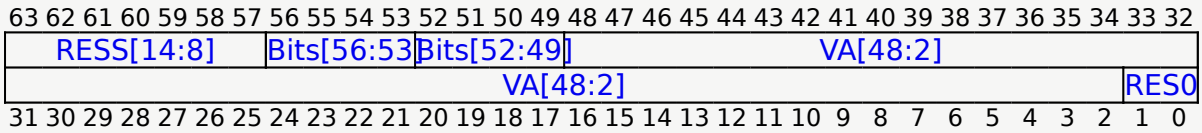
If breakpoint n is not implemented then accesses to this register are undefined.

Attributes

DBGBVR<n>_EL1 is a 64-bit register.

Field descriptions

When DBGBCR<n>_EL1.BT == 0b000x:



RESS[14:8], bits [63:57]

Reserved, Sign extended. Software must set all bits in this field to the same value as the most significant bit of the VA field. If all bits in this field are not the same value as the most significant bit of the VA field, then all of the following apply:

- It is constrained unpredictable whether the PE ignores this field when comparing an address.
- If the breakpoint is not context-aware, it is implementation defined whether the value read back in each bit of this field is a copy of the most significant bit of the VA field or the value written.

Bits[56:53]
When FEAT_LVA3 is implemented:

VA[56:53], bits [3:0] of bits [56:53]

Extension to VA[48:2]. For more information, see VA[48:2].

The reset behavior of this field is:

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

Otherwise:

RESS[7:4], bits [3:0] of bits [56:53]

Extension to RESS[14:8]. For more information, see RESS[14:8].

Bits[52:49]
When FEAT_LVA is implemented:

VA[52:49], bits [3:0] of bits [52:49]

Extension to VA[48:2]. For more information, see VA[48:2].

The reset behavior of this field is:

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

Otherwise:

RESS[3:0], bits [3:0] of bits [52:49]

Extension to RESS[14:8]. For more information, see RESS[14:8].

VA[48:2], bits [48:2]

Bits[48:2] of the address value for comparison.

When FEAT_LVA3 is implemented, (VA[56:53]:VA[52:49]) forms the upper part of the address value. If FEAT_LVA3 is not implemented, bits VA[56:53] are part of the RESS field.

When FEAT_LVA is implemented, VA[52:49] forms the upper part of the address value. If FEAT_LVA is not implemented, bits [52:49] are part of the RESS field.

The reset behavior of this field is:

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

Bits [1:0]

Reserved, res0.

When DBGBCR<n>_EL1.BT == 0b001x:

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
ContextID
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.

ContextID, bits [31:0]

Context ID value for comparison.

The value is compared against CONTEXTIDR_EL2 when (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented), HCR_EL2.E2H is 1, and either:

- The PE is executing at EL2.

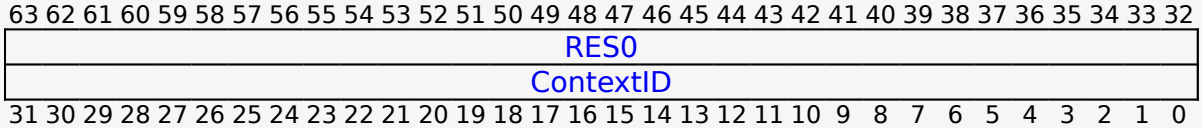
- HCR_EL2.TGE is 1, the PE is executing at EL0, and EL2 is enabled in the current Security state.

Otherwise, the value is compared against CONTEXTIDR_EL1.

The reset behavior of this field is:

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

When DBGBCR<n>_EL1.BT == 0b011x:



Bits [63:32]

Reserved, res0.

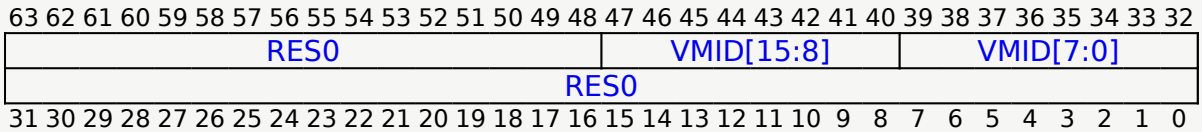
ContextID, bits [31:0]

Context ID value for comparison against CONTEXTIDR_EL1.

The reset behavior of this field is:

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

When DBGBCR<n>_EL1.BT == 0b100x and EL2 is implemented:



Bits [63:48]

Reserved, res0.

VMID[15:8], bits [47:40]
When FEAT_VMID16 is implemented, VTCR_EL2.VS == 1 and EL2 is using AArch64:

Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0].

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

VMID[7:0], bits [39:32]

VMID value for comparison.

The VMID is 8 bits when any of the following are true:

- EL2 is using AArch32.
- VTCR_EL2.VS is 0.
- FEAT_VMID16 is not implemented.

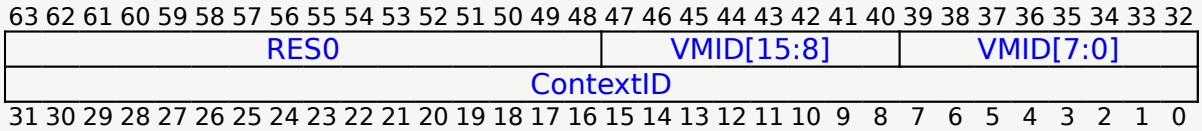
The reset behavior of this field is:

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

Bits [31:0]

Reserved, res0.

When DBGBCR<n>_EL1.BT == 0b101x and EL2 is implemented:



Bits [63:48]

Reserved, res0.

VMID[15:8], bits [47:40]
When FEAT_VMID16 is implemented, VTCR_EL2.VS == 1 and EL2 is using AArch64:

Extension to VMID[7:0]. For more information, see DBGBVR<n>_EL1.VMID[7:0].

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

VMID[7:0], bits [39:32]

VMID value for comparison.

The VMID is 8 bits when any of the following are true:

- EL2 is using AArch32.
- VTCR_EL2.VS is 0.
- FEAT_VMID16 is not implemented.

The reset behavior of this field is:

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

ContextID, bits [31:0]

Context ID value for comparison against CONTEXTIDR_EL1.

The reset behavior of this field is:

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

When DBGBCR<n>_EL1.BT == 0b110x, EL2 is implemented and (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented):

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
ContextID2
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

ContextID2, bits [63:32]

Context ID value for comparison against CONTEXTIDR_EL2.

The reset behavior of this field is:

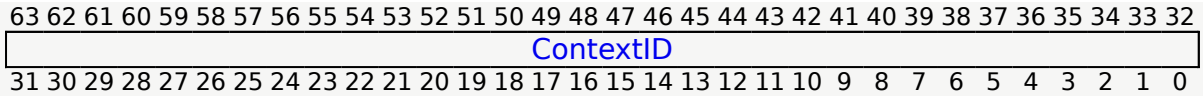
- On a Cold reset, this field resets to an architecturally unknown value.

Bits [31:0]

Reserved, res0.

When DBGBCR<n>_EL1.BT == 0b111x, EL2 is implemented and (FEAT_VHE is implemented or FEAT_Debugv8p2 is implemented):

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
ContextID2
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



ContextID2, bits [63:32]

Context ID value for comparison against CONTEXTIDR_EL2.

The reset behavior of this field is:

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

ContextID, bits [31:0]

Context ID value for comparison against CONTEXTIDR_EL1.

The reset behavior of this field is:

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

Accessing DBGBVR<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>, DBGBVR<m>_EL1 ; Where m = 0-15

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b100

```
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.DBGBVRn_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] = DBGBVR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)];
        else
            X[t, 64] = DBGBVR_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] = DBGBVR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)];
        else
            X[t, 64] = DBGBVR_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] = DBGBVR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)];
        else
            X[t, 64] = DBGBVR_EL1[m];
```

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

op0	op1	CRn	CRm	op2

0b10	0b000	0b0000	m[3:0]	0b100

```
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.DBGBVRn_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
            DBGBVR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)] = X[t, 64];
        else
            DBGBVR_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
            DBGBVR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)] = X[t, 64];
        else
            DBGBVR_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
            DBGBVR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)] = X[t, 64];
        else
            DBGBVR_EL1[m] = X[t, 64];
```

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94