DBGWCR<n>_EL1, Debug Watchpoint Control Registers, n = 0 - 63

The DBGWCR<n>_EL1 characteristics are:

Purpose

Holds control information for a watchpoint. Forms watchpoint n together with value register DBGWVR<n>_EL1.

Configuration

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

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

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

Attributes

DBGWCR<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					RES0				WT	LBN				SSC	HMC	BAS									LSC		PAC		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 data address watchpoints, with DBGWCR<n>_EL1.LBN, specifies the index of the breakpoint linked to.

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

This field extends DBGWCR<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 watchpoint can be generated are: HMC, PAC, 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]

Address Mask. Only objects 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 watchpoint behaves as if either:

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

The reset behavior of this field is:

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

Bits [23:21]

Reserved, res0.

WT, bit [20]

Watchpoint type. Possible values are:

WT	Meaning
0b0	Unlinked data address match.
0b1	Linked data address match.

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 data address watchpoints, with DBGWCR<n>_EL1.LBNX when implemented, specifies the index of the breakpoint linked to.

For all other watchpoint 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 Watchpoint debug event for watchpoint n is generated.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, 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 watchpoint generates Watchpoint exceptions'.

For more information on the effect of programming the fields to a reserved value, see 'Reserved DBGWCR<n>_EL1.{SSC, HMC, PAC} 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 Watchpoint debug event for watchpoint n is generated.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, 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 watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

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

BAS, bits [12:5]

Byte address select. Each bit of this field selects whether a byte from within the word or double-word addressed by DBGWVR<n>_EL1 is being watched.

BAS	Description
xxxxxxx1	Match byte at DBGWVR<n>_EL1
xxxxxx1x	Match byte at DBGWVR<n>_EL1 + 1
xxxxx1xx	Match byte at DBGWVR<n>_EL1 + 2
xxxx1xxx	Match byte at DBGWVR<n>_EL1 + 3

In cases where DBGWVR<n>_EL1 addresses a double-word:

BAS	Description, if DBGWVR<n>_EL1[2] == 0
xxx1xxxx	Match byte at DBGWVR<n>_EL1 + 4
xx1xxxxx	Match byte at DBGWVR<n>_EL1 + 5
x1xxxxxx	Match byte at DBGWVR<n>_EL1 + 6
1xxxxxxx	Match byte at DBGWVR<n>_EL1 + 7

If DBGWVR<n>_EL1[2] == 1, only BAS[3:0] are used and BAS[7:4] are ignored. Arm deprecates setting DBGWVR<n>_EL1[2] == 1.

The valid values for BAS are nonzero binary numbers all of whose set bits are contiguous. All other values are reserved and must not be used by software. See 'Reserved DBGWCR<n>_EL1.BAS values'.

The reset behavior of this field is:

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

LSC, bits [4:3]

Load/store control. This field enables watchpoint matching on the type of access being made. Possible values of this field are:

LSC	Meaning
0b01	Match instructions that load from a watchpointed address.
0b10	Match instructions that store to a watchpointed address.
0b11	Match instructions that load from or store to a watchpointed address.

All other values are reserved, but must behave as if the watchpoint is disabled. Software must not rely on this property as the behavior of reserved values might change in a future revision of the architecture.

The reset behavior of this field is:

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

PAC, bits [2:1]

Privilege of access control. Determines the Exception level or levels at which a Watchpoint debug event for watchpoint n is generated.

The fields that indicate when the watchpoint can be generated are: HMC, PAC, 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 watchpoint generates Watchpoint exceptions'.

The reset behavior of this field is:

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

E, bit [0]

Enable watchpoint n.

E	Meaning
0b0	Watchpoint n disabled.
0b1	Watchpoint 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 DBGWCR<n>_EL1

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

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

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

op0	op1	CRn	CRm	op2

0b10	0b000	0b0000	m[3:0]	0b111

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

if (!IsFeatureImplemented(FEAT_Debugv8p9) && m >=
NUM_WATCHPOINTS) ||
(IsFeatureImplemented(FEAT_Debugv8p9) && m +
(UInt(MDSELR_EL1.BANK) * 16) >= NUM_WATCHPOINTS) 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.DBGWCRn_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] = DBGWCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)];
        else
            X[t, 64] = DBGWCR_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] = DBGWCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)];
        else
            X[t, 64] = DBGWCR_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] = DBGWCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)];
        else
            X[t, 64] = DBGWCR_EL1[m];
```

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

op0	op1	CRn	CRm	op2
0b10	0b000	0b0000	m[3:0]	0b111

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

if (!IsFeatureImplemented(FEAT_Debugv8p9) && m >=
NUM_WATCHPOINTS) ||
(IsFeatureImplemented(FEAT_Debugv8p9) && m +
(UInt(MDSELR_EL1.BANK) * 16) >= NUM_WATCHPOINTS) 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.DBGWCRn_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
            DBGWCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)] = X[t, 64];
        else
            DBGWCR_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
            DBGWCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)] = X[t, 64];
        else
            DBGWCR_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
            DBGWCR_EL1[m +
(UInt(EffectiveMDSELR_EL1_BANK()) * 16)] = X[t, 64];
        else
            DBGWCR_EL1[m] = X[t, 64];
```

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94