

ERXPFGF_EL1, Selected Pseudo-fault Generation Feature Register

The ERXPFGF_EL1 characteristics are:

Purpose

Accesses [ERR<n>PFGF](#) for the error record <n> selected by [ERRSELR_EL1](#).SEL.

Configuration

This register is present only when FEAT_RASv1p1 is implemented. Otherwise, direct accesses to ERXPFGF_EL1 are undefined.

Attributes

ERXPFGF_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
ERR<n>PFGF																															
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
ERR<n>PFGF																															

Bits [63:0]

ERXPFGF_EL1 accesses [ERR<n>PFGF](#), where <n> is the value in [ERRSELR_EL1](#).SEL.

Accessing ERXPFGF_EL1

If [ERRIDR_EL1](#).NUM is 0x0000 or [ERRSELR_EL1](#).SEL is greater than or equal to [ERRIDR_EL1](#).NUM, then one of the following occurs:

- An unknown error record is selected.
- ERXPFGF_EL1 is RAZ.
- Direct reads of ERXPFGF_EL1 are NOPs.
- Direct reads of ERXPFGF_EL1 are undefined.

If [ERRSELR_EL1](#).SEL selects an error record owned by a node that does not implement the Common Fault Injection Model Extension, then one of the following occurs:

- ERXPFGF_EL1 is RAZ.

- Direct reads of `ERXPFGF_EL1` are NOPs.
- Direct reads of `ERXPFGF_EL1` are undefined.

Note

A node does not implement the Common Fault Injection Model Extension if `ERR<q>FR.INJ` reads as `0b00`. `<q>` is the index of the first error record owned by the same node as error record `<n>`, where `<n>` is the value in `ERRSELR_EL1.SEL`. If the node owns a single record then `q = n`.

If `ERRSELR_EL1.SEL` is not the index of the first error record owned by a node, then `ERR<n>PFGF` is not present, meaning reads of `ERXPFGF_EL1` are `res0`.

`ERR<n>PFGF` describes additional constraints that also apply when `ERR<n>PFGF` is accessed through `ERXPFGF_EL1`.

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

MRS <Xt>, ERXPFGF_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0101	0b0100	0b100

```

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'" && SCR_EL3.FIEN == '0' then
        UNDEFINED;
    elsif EL2Enabled() && HCR_EL2.FIEN == '0' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif EL2Enabled() &&
    IsFeatureImplemented(FEAT_FGT) && (!HaveEL(EL3) ||
    SCR_EL3.FGTEn == '1') && HFGTR_EL2.ERXPFGF_EL1 ==
    '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x18);
        else
            X[t, 64] = ERXPFGF_EL1;
    elsif PSTATE.EL == EL2 then

```

```

        if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
        && boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && SCR_EL3.FIEN == '0' then
            UNDEFINED;
        elsif HaveEL(EL3) && SCR_EL3.FIEN == '0' then
            if Halted() && EDSCR.SDD == '1' then
                UNDEFINED;
            else
                AArch64.SystemAccessTrap(EL3, 0x18);
            else
                X[t, 64] = ERXPFGEF_EL1;
        elsif PSTATE.EL == EL3 then
            X[t, 64] = ERXPFGEF_EL1;

```

[AArch32
Registers](#)

[AArch64
Registers](#)

[AArch32
Instructions](#)

[AArch64
Instructions](#)

[Index by
Encoding](#)

[External
Registers](#)

28/03/2023 16:02; 72747e43966d6b97dcbd230a1b3f0421d1ea3d94

Copyright © 2010-2023 Arm Limited or its affiliates. All rights reserved. This document is Non-Confidential.