FPCR, Floating-point Control Register

The FPCR characteristics are:

Purpose

Controls floating-point behavior.

Configuration

AArch64 System register FPCR bits [26:15] are architecturally mapped to AArch32 System register FPSCR[26:15].

AArch64 System register FPCR bits [12:8] are architecturally mapped to AArch32 System register FPSCR[12:8].

It is implementation defined whether the Len and Stride fields can be programmed to nonzero values, which will cause some AArch32 floating-point instruction encodings to be undefined, or whether these fields are RAZ.

Attributes

FPCR 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																					
RES0	AHP	DN	FZ	RMode		Stride		FZ16	Len	IDE	RES0	EBF	IXE	UFE	OFE	DZE	IOE	RES0	NEP	AH	FIZ
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 [63:27]

Reserved, res0.

AHP, bit [26]

Alternative half-precision control bit.

AHP	Meaning
`0b0`	IEEE half-precision format selected.
`0b1`	Alternative half-precision format selected.

This bit is used only for conversions between half-precision floating-point and other floating-point formats.

The data-processing instructions added as part of the FEAT_FP16 extension always use the IEEE half-precision format, and ignore the value of this bit.

The reset behavior of this field is:

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

DN, bit [25]

Default NaN use for NaN propagation.

DN	Meaning
`0b0`	NaN operands propagate through to the output of a floating-point operation.
`0b1`	Any operation involving one or more NaNs returns the Default NaN. This bit has no effect on the output of FABS, FMAX*, FMIN*, and FNEG instructions, and a default NaN is never returned as a result of these instructions.

The value of this bit controls both scalar and Advanced SIMD floating-point arithmetic.

The reset behavior of this field is:

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

FZ, bit [24]

Flushing denormalized numbers to zero control bit.

FZ	Meaning

`0b0`	If FPCR.AH is 0, the flushing to zero of single-precision and double-precision denormalized inputs to, and outputs of, floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero. If FPCR.AH is 1, the flushing to zero of single-precision and double-precision denormalized outputs of floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero.
`0b1`	If FPCR.AH is 0, denormalized single-precision and double-precision inputs to, and outputs from, floating-point instructions are flushed to zero. If FPCR.AH is 1, denormalized single-precision and double-precision outputs from floating-point instructions are flushed to zero.

For more information, see 'Flushing denormalized numbers to zero' and the pseudocode of the floating-point instructions.

The reset behavior of this field is:

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

RMode, bits [23:22]

Rounding Mode control field.

RMode	Meaning
`0b00`	Round to Nearest (RN) mode.
`0b01`	Round towards Plus Infinity (RP) mode.
`0b10`	Round towards Minus Infinity (RM) mode.
`0b11`	Round towards Zero (RZ) mode.

The specified rounding mode is used by both scalar and Advanced SIMD floating-point instructions.

If FPCR.AH is 1, then the following instructions use Round to Nearest mode regardless of the value of this bit:

- The FRECPE, FRECPS, FRECPX, FRSQRTE, and FRSQRTS instructions.
- The BFCVT, BFCVTN, BFCVTN2, BFCVTNT, BFMLALB, and BFMLALT instructions.

The reset behavior of this field is:

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

Stride, bits [21:20]

This field has no function in AArch64 state, and nonzero values are ignored during execution in AArch64 state.

This field is included only for context saving and restoration of the AArch32 FPSCR.Stride field.

The reset behavior of this field is:

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

FZ16, bit [19]
When FEAT_FP16 is implemented:

Flushing denormalized numbers to zero control bit on half-precision data-processing instructions.

FZ16	Meaning
`0b0`	For some instructions, this bit disables flushing to zero of inputs and outputs that are half-precision denormalized numbers.
`0b1`	Flushing denormalized numbers to zero enabled. For some instructions that do not convert a half-precision input to a higher precision output, this bit enables flushing to zero of inputs and outputs that are half-precision denormalized numbers.

The value of this bit applies to both scalar and Advanced SIMD floating-point half-precision calculations.

For more information, see 'Flushing denormalized numbers to zero' and the pseudocode of the floating-point instructions.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

Len, bits [18:16]

This field has no function in AArch64 state, and nonzero values are ignored during execution in AArch64 state.

This field is included only for context saving and restoration of the AArch32 FPSCR.Len field.

The reset behavior of this field is:

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

IDE, bit [15]

Input Denormal floating-point exception trap enable.

IDE	Meaning
`0b0`	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR.IDC bit is set to 1.
`0b1`	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR.IDC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled, the value of FPCR.IDE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The Effective value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

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

Bit [14]

Reserved, res0.

EBF, bit [13]
When FEAT_EBF16 is implemented:

The value of this bit controls the numeric behaviors of BFloat16 dot product calculations performed by the BFDOT, BFMMLA, BFMOPA, and BFMOPS instructions. If FEAT_SME2 is implemented, this also controls BFVDOT instruction.

When ID_AA64ISAR1_EL1.BF16 and ID_AA64ZFR0_EL1.BF16 are `0b0010`, the PE supports the FPCR.EBF field. Otherwise, FPCR.EBF is res0.

EBF	Meaning
`0b0`	These instructions use the standard BFloat16 behaviors: • Ignoring the FPCR.RMode control and using the rounding mode defined for BFloat16. For more information, see 'Round to Odd mode'. • Flushing denormalized inputs and outputs to zero, as if the FPCR.FZ and FPCR.FIZ controls had the value '1'. • Performing unfused multiplies and additions with intermediate rounding of all products and sums.

`0b1`	These instructions use the extended BFloat16 behaviors: • Supporting all four IEEE 754 rounding modes selected by the FPCR.RMode control. • Optionally, flushing denormalized inputs and outputs to zero, as governed by the FPCR.FZ and FPCR.FIZ controls. • Performing a fused two-way sum-of-products for each pair of adjacent BFloat16 elements, without intermediate rounding of the products, but rounding the single-precision sum before addition to the accumulator. • Generating the default NaN as intermediate sum-of-products when any multiplier input is a NaN, or any product is infinity Ã— 0.0, or there are infinite products with differing signs. • Generating an intermediate sum-of-products of the same infinity when there are infinite products all with the same sign.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

IXE, bit [12]

Inexact floating-point exception trap enable.

IXE	Meaning

`0b0`	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR.IXC bit is set to 1.
`0b1`	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR.IXC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled, the value of FPCR.IXE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The Effective value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

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

UFE, bit [11]

Underflow floating-point exception trap enable.

UFE	Meaning
`0b0`	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR.UFC bit is set to 1.
`0b1`	Trapped exception handling selected. If the floating-point exception occurs and Flush-to-zero is not enabled, the PE does not update the FPSR.UFC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled, the value of FPCR.UFE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The Effective value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

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

OFE, bit [10]

Overflow floating-point exception trap enable.

OFE	Meaning
0b0	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR.OFC bit is set to 1.
0b1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR.OFC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled, the value of FPCR.OFE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The Effective value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

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

DZE, bit [9]

Divide by Zero floating-point exception trap enable.

DZE	Meaning
0b0	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR.DZC bit is set to 1.
0b1	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR.DZC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled, the value of FPCR.DZE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The Effective value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

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

IOE, bit [8]

Invalid Operation floating-point exception trap enable.

IOE	Meaning
`0b0`	Untrapped exception handling selected. If the floating-point exception occurs, the FPSR.IOC bit is set to 1.
`0b1`	Trapped exception handling selected. If the floating-point exception occurs, the PE does not update the FPSR.IOC bit.

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled, the value of FPCR.IOE is treated as 0 for all purposes other than a direct read or write of the FPCR.

The Effective value of this bit controls both scalar and vector floating-point arithmetic.

If the implementation does not support this exception, this bit is RAZ/WI.

The reset behavior of this field is:

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

Bits [7:3]

Reserved, res0.

NEP, bit [2]
When FEAT_AFP is implemented:

Controls how the output elements other than the lowest element of the vector are determined for Advanced SIMD scalar instructions.

NEP	Meaning

`0b0`	Does not affect how the output elements other than the lowest are determined for Advanced SIMD scalar instructions.

`0b1`	The output elements other than the lowest are taken from the following registers: • For 3-input scalar versions of the FMLA (by element) and FMLS (by element) instructions, the <Hd>, <Sd>, or <Dd> register. • For 3-input versions of the FMADD, FMSUB, FNMADD, and FNMSUB instructions, the <Ha>, <Sa>, or <Da> register. • For 2-input scalar versions of the FACGE, FACGT, FCMEQ (register), FCMGE (register), and FCMGT (register) instructions, the <Hm>, <Sm>, or <Dm> register. • For 2-input scalar versions of the FABD, FADD (scalar), FDIV (scalar), FMAX (scalar), FMAXNM (scalar), FMIN (scalar), FMINNM (scalar), FMUL (by element), FMUL (scalar), FMULX (by element), FMULX, FNMUL (scalar), FRECPS, FRSQRTS, and FSUB (scalar) instructions, the <Hn>, <Sn>, or <Dn> register. • For 1-input scalar versions of the following instructions, the <Hd>, <Sd>, or <Dd> register: ◦ The (vector) versions of the FCVTAS, FCVTAU, FCVTMS, FCVTMU, FCVTNS, FCVTNU, FCVTPS, and FCVTPU instructions. ◦ The (vector, fixed-point) and (vector, integer) versions of the FCVTZS, FCVTZU, SCVTF, and UCVTF instructions. ◦ The (scalar) versions of the FABS, FNEG, FRINT32X, FRINT32Z, FRINT64X, FRINT64Z, FRINTA, FRINTI, FRINTM, FRINTN, FRINTP, FRINTX, FRINTZ, and FSQRT

When the PE is in Streaming SVE mode, and FEAT_SME_FA64 is not implemented or not enabled, the value of FPCR.NEP is treated as 0 for all purposes other than a direct read or write of the FPCR.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

AH, bit [1]
When FEAT_AFP is implemented:

Alternate Handling. Controls alternate handling of floating-point numbers.

The Arm architecture supports two models for handling some of the corner cases of the floating-point behaviors, such as the nature of flushing of denormalized numbers, the detection of tininess and other exceptions and a range of other behaviors. The value of the FPCR.AH bit selects between these models.

For more information on the FPCR.AH bit, see 'Flushing denormalized numbers to zero', 'Floating-point exceptions and exception traps' and the pseudocode of the floating-point instructions.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

FIZ, bit [0]
When FEAT_AFP is implemented:

Flush Inputs to Zero. Controls whether single-precision, double-precision and BFloat16 input operands that are denormalized numbers are flushed to zero.

FIZ	Meaning

0b0	The flushing to zero of single-precision and double-precision denormalized inputs to floating-point instructions not enabled by this control, but other factors might cause the input denormalized numbers to be flushed to zero.
0b1	Denormalized single-precision and double-precision inputs to most floating-point instructions flushed to zero.

For more information, see 'Flushing denormalized numbers to zero' and the pseudocode of the floating-point instructions.

The reset behavior of this field is:

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

Otherwise:

Reserved, res0.

Accessing FPCR

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

MRS <Xt>, FPCR

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b000

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> ==
'11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& CPTR_EL2.FPEN != '11' then
```

```
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' &&
CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' &&
CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN ==
'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        X[t, 64] = FPCR;
```

MSR FPCR, <Xt>

op0	op1	CRn	CRm	op2
0b11	0b011	0b0100	0b0100	0b000

```
if PSTATE.EL == EL0 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif !(EL2Enabled() && HCR_EL2.<E2H,TGE> ==
'11') && CPACR_EL1.FPEN != '11' then
        if EL2Enabled() && HCR_EL2.TGE == '1' then
            AArch64.SystemAccessTrap(EL2, 0x00);
        else
            AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.<E2H,TGE> == '11'
&& CPTR_EL2.FPEN != '11' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' &&
CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL1 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
        UNDEFINED;
    elsif CPACR_EL1.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL1, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H != '1' &&
CPTR_EL2.TFP == '1' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif EL2Enabled() && HCR_EL2.E2H == '1' &&
CPTR_EL2.FPEN == 'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL2 then
    if Halted() && HaveEL(EL3) && EDSCR.SDD == '1'
&& boolean IMPLEMENTATION_DEFINED "EL3 trap priority
when SDD == '1'" && CPTR_EL3.TFP == '1' then
```

```
        UNDEFINED;
    elsif HCR_EL2.E2H == '0' && CPTR_EL2.TFP == '1'
then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HCR_EL2.E2H == '1' && CPTR_EL2.FPEN ==
'x0' then
        AArch64.SystemAccessTrap(EL2, 0x07);
    elsif HaveEL(EL3) && CPTR_EL3.TFP == '1' then
        if Halted() && EDSCR.SDD == '1' then
            UNDEFINED;
        else
            AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
elsif PSTATE.EL == EL3 then
    if CPTR_EL3.TFP == '1' then
        AArch64.SystemAccessTrap(EL3, 0x07);
    else
        FPCR = X[t, 64];
```