Arm® Cortex®-A35 Processor Advanced SIMD and Floating-point Support

Revision: r1p0

Technical Reference Manual

arm

100238_0100_00_en

Arm® Cortex®-A35 Processor Advanced SIMD and Floating-point Support

Technical Reference Manual

Release Information

Document History

Issue	Date	Confidentiality	Change
0000-00	04 December 2015	Confidential	First release for r0p0
0001-00	18 March 2016	Confidential	First release for r0p1
0002-00	04 March 2017	Non-Confidential	First release for r0p2
0100-00	28 February 2019	Non-Confidential	First release for r1p0

Non-Confidential Proprietary Notice

LES-PRE-20349

Confidentiality Status

This document is Non-Confidential. The right to use, copy and disclose this document may be subject to license restrictions in accordance with the terms of the agreement entered into by Arm and the party that Arm delivered this document to.

Unrestricted Access is an Arm internal classification.

Product Status

The information in this document is Final, that is for a developed product.

Web Address

http://www.arm.com

Contents

Arm® Cortex®-A35 Processor Advanced SIMD and Floating-point Support Technical Reference Manual

Preface

This preface introduces the *Arm® Cortex®-A35 Processor Advanced SIMD and Floating-point Support Technical Reference Manual*.

It contains the following:

About this book

This book is for the Cortex®-A35 processor Advanced SIMD and floating-point support.

Product revision status

The r*m*p*n* identifier indicates the revision status of the product described in this book, for example, r*1*p*2*, where:

r*m* Identifies the major revision of the product, for example, r1.

p*n* Identifies the minor revision or modification status of the product, for example, p2.

Intended audience

This manual is written for system designers, system integrators, and programmers who are designing or programming a System-on-Chip (SoC) that uses the Cortex®-A35 processor with the optional Advanced SIMD and floating-point support.

Using this book

This book is organized into the following chapters:

Chapter 1 Functional Description
This chapter introduces the optional Advanced SIMD and floating-point support.

Chapter 2 AArch64 Register Descriptions
This chapter describes the AArch64 registers for the Cortex-A35 processor Advanced SIMD and floating-point support.

Chapter 3 AArch32 Register Descriptions
This chapter describes the AArch32 registers for the Cortex-A35 processor Advanced SIMD and floating-point support.

Appendix A Revisions
This appendix describes the technical changes between released issues of this book.

Glossary

The Arm® Glossary is a list of terms used in Arm documentation, together with definitions for those terms. The Arm Glossary does not contain terms that are industry standard unless the Arm meaning differs from the generally accepted meaning.

See the *Arm® Glossary* for more information.

Typographic conventions

italic
Introduces special terminology, denotes cross-references, and citations.

bold
Highlights interface elements, such as menu names. Denotes signal names. Also used for terms in descriptive lists, where appropriate.

`monospace`
Denotes text that you can enter at the keyboard, such as commands, file and program names, and source code.

`mono`space
Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

`monospace italic`
Denotes arguments to monospace text where the argument is to be replaced by a specific value.

`monospace bold`
Denotes language keywords when used outside example code.

`<and>`
Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example:

```
MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>
```

SMALL CAPITALS
Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

Timing diagrams

The following figure explains the components used in timing diagrams. Variations, when they occur, have clear labels. You must not assume any timing information that is not explicit in the diagrams.

Shaded bus and signal areas are undefined, so the bus or signal can assume any value within the shaded area at that time. The actual level is unimportant and does not affect normal operation.

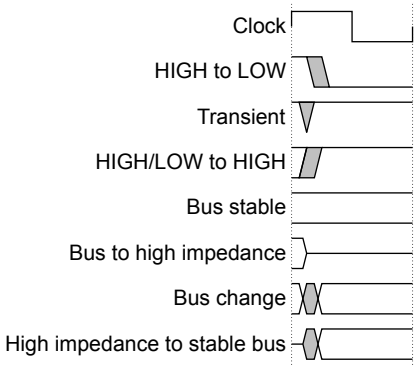

Figure 1 Key to timing diagram conventions

Signals

The signal conventions are:

Signal level
The level of an asserted signal depends on whether the signal is active-HIGH or active-LOW. Asserted means:
- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

At the start or end of a signal name denotes an active-LOW signal.

Additional reading

This book contains information that is specific to this product. See the following documents for other relevant information.

Arm publications
- *Arm® Cortex®-A35 Processor Technical Reference Manual* (100236).
- *Arm® Cortex®-A35 Processor Integration Manual* (100240).
- *Arm® Cortex®-A35 Configuration and Sign-off Guide* (100239).
- *Arm® Cortex®-A35 Processor Cryptographic Extension Technical Reference Manual* (100237).
- *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* (DDI 0487).

Other publications

- ANSI/IEEE Std 754-2008, IEEE Standard for Floating-Point Arithmetic.

Feedback

Feedback on this product

If you have any comments or suggestions about this product, contact your supplier and give:

- The product name.
- The product revision or version.
- An explanation with as much information as you can provide. Include symptoms and diagnostic procedures if appropriate.

Feedback on content

If you have comments on content then send an e-mail to *errata@arm.com*. Give:

- The title *Arm Cortex-A35 Processor Advanced SIMD and Floating-point Support Technical Reference Manual*.
- The number 100238_0100_00_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

Note

Arm tests the PDF only in Adobe Acrobat and Acrobat Reader, and cannot guarantee the quality of the represented document when used with any other PDF reader.

Chapter 1
Functional Description

This chapter introduces the optional Advanced SIMD and floating-point support.

It contains the following section:

1.1 About the Advanced SIMD and floating-point support

The Cortex-A35 processor supports the Advanced SIMD and scalar floating-point instructions in the A64 instruction set and the Advanced SIMD and floating-point instructions in the A32 and T32 instruction sets.

The Cortex-A35 floating-point implementation:

- Does not support floating-point exception trapping.
- Implements all scalar operations in hardware with support for all combinations of:
  - Rounding modes.
  - Flush-to-zero.
  - Default *Not a Number* (NaN) modes.

The Armv8 architecture eliminates the concept of version numbers for its Advanced SIMD and floating-point support in the AArch64 execution state.

Chapter 2
AArch64 Register Descriptions

This chapter describes the AArch64 registers for the Cortex-A35 processor Advanced SIMD and floating-point support.

It contains the following sections:

2.1 Accessing the AArch64 feature identification registers

Software can identify the Advanced SIMD and floating-point features using the feature identification registers in the AArch64 execution state.

You can access the feature identification registers in the AArch64 execution state using the `MRS` instruction, for example:

```
MRS <Xt>, ID_AA64PFR0_EL1 ; Read ID_AA64PFR0_EL1 into Xt
MRS <Xt>, MVFR0_EL1       ; Read MVFR0_EL1 into Xt
MRS <Xt>, MVFR1_EL1       ; Read MVFR1_EL1 into Xt
MRS <Xt>, MVFR2_EL1       ; Read MVFR2_EL1 into Xt
```

Table 2-1 AArch64 Advanced SIMD and scalar floating-point feature identification registers

AArch64 name	Description
ID_AA64PFR0_EL1	Gives additional information about implemented processor features in AArch64. See the *Arm® Cortex®-A35 Processor Technical Reference Manual*.
MVFR0_EL1	See *2.5 Media and VFP Feature Register 0* on page 2-20.
MVFR1_EL1	See *2.6 Media and VFP Feature Register 1* on page 2-22.
MVFR2_EL1	See *2.7 Media and VFP Feature Register 2* on page 2-24.

2.2 AArch64 register summary

The processor has several Advanced SIMD and floating-point system registers in the AArch64 execution state. Each register has a specific purpose, specific usage constraints, configurations, and attributes.

The following table gives a summary of the Cortex-A35 processor Advanced SIMD and floating-point system registers in the AArch64 execution state.

Table 2-2 AArch64 Advanced SIMD and floating-point system registers

Name	Type	Reset	Description
FPCR	RW	0x00000000	See *2.3 Floating-point Control Register* on page 2-16.
FPSR	RW	0x00000000	See *2.4 Floating-point Status Register* on page 2-18.
MVFR0_EL1	RO	0x10110222	See *2.5 Media and VFP Feature Register 0* on page 2-20.
MVFR1_EL1	RO	0x12111111	See *2.6 Media and VFP Feature Register 1* on page 2-22.
MVFR2_EL1	RW	0x00000043	See *2.7 Media and VFP Feature Register 2* on page 2-24.
FPEXC32_EL2	RW	0x00000700	See *2.8 Floating-point Exception Control Register* on page 2-26

2.3 Floating-point Control Register

The FPCR characteristics are:

Purpose

Controls floating-point extension behavior.

Usage constraints

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
RW	RW	RW	RW	RW	RW

Configurations

The named fields in this register map to the equivalent fields in the AArch32 FPSCR. See *3.4 Floating-Point Status and Control Register* on page 3-33.

Attributes

FPCR is a 32-bit register.

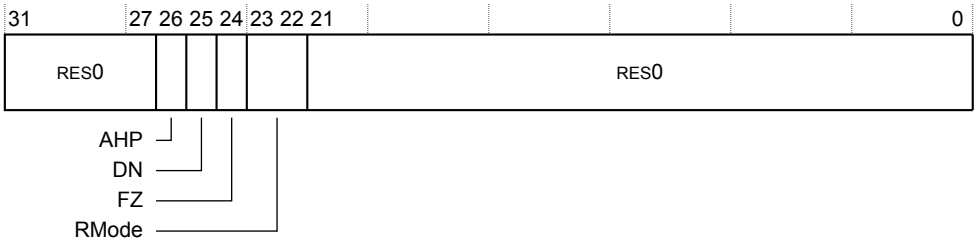

Figure 2-1 FPCR bit assignments

[31:27]

Reserved, RES0.

AHP, [26]

Alternative half-precision control bit. The possible values are:

`0` IEEE half-precision format selected. This is the reset value.

`1` Alternative half-precision format selected.

DN, [25]

Default NaN mode control bit. The possible values are:

`0` NaN operands propagate through to the output of a floating-point operation. This is the reset value.

`1` Any operation involving one or more NaNs returns the Default NaN.

FZ, [24]

Flush-to-zero mode control bit. The possible values are:

`0` Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the reset value.

`1` Flush-to-zero mode enabled.

RMode, [23:22]

Rounding Mode control field. The encoding of this field is:

`0b00` *Round to Nearest* (RN) mode. This is the reset value.

`0b01` *Round towards Plus Infinity* (RP) mode.

`0b10` *Round towards Minus Infinity* (RM) mode.

`0b11` *Round towards Zero* (RZ) mode.

[21:0]

Reserved, RES0.

To access the FPCR:

```
MRS <Xt>, FPCR ; Read FPCR into XtMSR FPCR, <Xt> ; Write Xt to FPCR
```

Table 2-3 FPCR access encoding

op0	op1	CRn	CRm	op2
11	011	0100	0100	000

2.4 Floating-point Status Register

The FPSR characteristics are:

Purpose

Provides floating-point system status information.

Usage constraints

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
RW	RW	RW	RW	RW	RW

Configurations

The named fields in this register map to the equivalent fields in the AArch32 FPSCR. See *3.4 Floating-Point Status and Control Register* on page 3-33.

Attributes

FPSR is a 32-bit register.

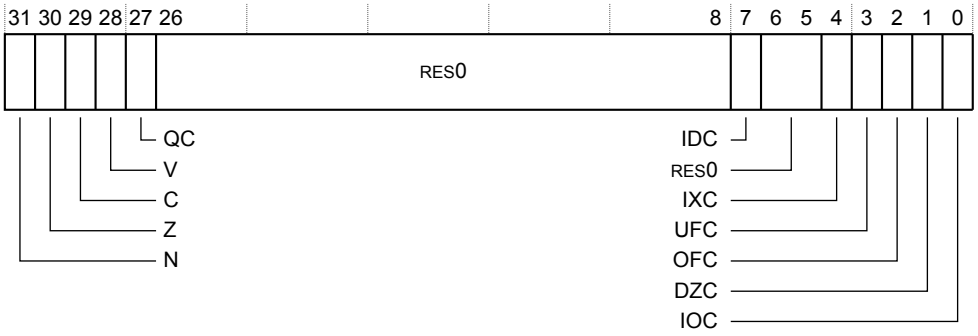

Figure 2-2 FPSR bit assignments

N, [31]

Negative condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.N flag instead.

Z, [30]

Zero condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.Z flag instead.

C, [29]

Carry condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.C flag instead

V, [28]

Overflow condition flag for AArch32 floating-point comparison operations. AArch64 floating-point comparisons set the PSTATE.V flag instead.

QC, [27]

Cumulative saturation bit. This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated since a 0 was last written to this bit.

[26:8]

Reserved, RES0.

IDC, [7]

Input Denormal cumulative exception bit. This bit is set to 1 to indicate that the Input Denormal exception has occurred since 0 was last written to this bit.

[6:5]

Reserved, RES0.

IXC, [4]

Inexact cumulative exception bit. This bit is set to 1 to indicate that the Inexact exception has occurred since 0 was last written to this bit.

UFC, [3]

Underflow cumulative exception bit. This bit is set to 1 to indicate that the Underflow exception has occurred since 0 was last written to this bit.

OFC, [2]

Overflow cumulative exception bit. This bit is set to 1 to indicate that the Overflow exception has occurred since 0 was last written to this bit.

DZC, [1]

Division by Zero cumulative exception bit. This bit is set to 1 to indicate that the Division by Zero exception has occurred since 0 was last written to this bit.

IOC, [0]

Invalid Operation cumulative exception bit. This bit is set to 1 to indicate that the Invalid Operation exception has occurred since 0 was last written to this bit.

To access the FPSR:

```
MRS <Xt>, FPSR; Read FPSR into Xt
MSR FPSR, <Xt>; Write Xt to FPSR
```

Table 2-4 FPSR access encoding

op0	op1	CRn	CRm	op2
11	011	0100	0100	001

2.5 Media and VFP Feature Register 0

The MVFR0_EL1 characteristics are:

Purpose

Describes the features provided by the AArch32 Advanced SIMD and floating-point implementation.

Usage constraints

This register is accessible as follows:

EL0	EL1(NS)	EL1(S)	EL2	EL3 (SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

Configurations

MVFR0_EL1 is architecturally mapped to AArch32 register MVFR0. See *3.5 Media and VFP Feature Register 0* on page 3-36.

Attributes

MVFR0_EL1 is a 32-bit register.

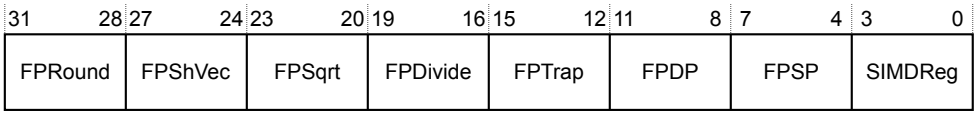

Figure 2-3 MVFR0_EL1 bit assignments

FPRound, [31:28]

Indicates the rounding modes supported by the floating-point hardware:

`0x1` All rounding modes supported.

FPShVec, [27:24]

Indicates the hardware support for floating-point short vectors:

`0x0` Not supported.

FPSqrt, [23:20]

Indicates the hardware support for floating-point square root operations:

`0x1` Supported.

FPDivide, [19:16]

Indicates the hardware support for floating-point divide operations:

`0x1` Supported.

FPTrap, [15:12]

Indicates whether the floating-point hardware implementation supports exception trapping:

`0x0` Not supported.

FPDP, [11:8]

Indicates the hardware support for floating-point double-precision operations:

`0x2` Supported, VFPv3 or greater.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

FPSP, [7:4]

Indicates the hardware support for floating-point single-precision operations:

`0x2` Supported, VFPv3 or greater.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

SIMDReg, [3:0]

Indicates support for the Advanced SIMD register bank:

`0x2` Supported, 32 x 64-bit registers supported.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

To access the MVFR0_EL1:

```
MRS <Xt>, MVFR0_EL1 ; Read MVFR0_EL1 into Xt
```

The following table shows the register access encoding.

Table 2-5 MVFR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	000

2.6 Media and VFP Feature Register 1

The MVFR1_EL1 characteristics are:

Purpose

Describes the features provided by the AArch32 Advanced SIMD and floating-point implementation.

Usage constraints

This register is accessible as follows:

EL0	EL1(NS)	EL1(S)	EL2	EL3 (SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

Configurations

MVFR1_EL1 is architecturally mapped to AArch32 register MVFR1. See *3.6 Media and VFP Feature Register 1* on page 3-38.

Attributes

MVFR1_EL1 is a 32-bit register.

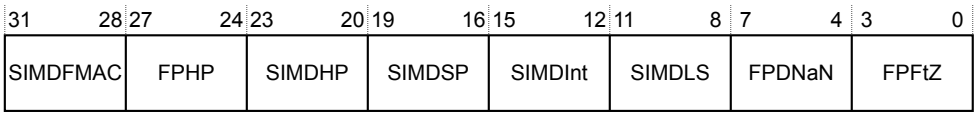

Figure 2-4 MVFR1_EL1 bit assignments

SIMDFMAC, [31:28]

Indicates whether the Advanced SIMD and floating-point unit supports fused multiply accumulate operations:

`0x1` Implemented.

FPHP, [27:24]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion instructions:

`0x2` Instructions to convert between half-precision and single-precision, and between half-precision and double-precision, implemented.

SIMDHP, [23:20]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion operations:

`0x1` Implemented.

SIMDSP, [19:16]

Indicates whether the Advanced SIMD and floating-point unit supports single-precision floating-point operations:

`0x1` Implemented.

SIMDInt, [15:12]

Indicates whether the Advanced SIMD and floating-point unit supports integer operations:

`0x1` Implemented.

SIMDLS, [11:8]

Indicates whether the Advanced SIMD and floating-point unit supports load/store instructions:

`0x1` Implemented.

FPDNaN, [7:4]

Indicates whether the floating-point hardware implementation supports only the Default NaN mode:

`0x1` Hardware supports propagation of NaN values.

FPFtZ, [3:0]

Indicates whether the floating-point hardware implementation supports only the Flush-to-Zero mode of operation:

`0x1` Hardware supports full denormalized number arithmetic.

To access the MVFR1_EL1:

```
MRS <Xt>, MVFR1_EL1 ; Read MVFR1_EL1 into Xt
```

The following table shows the register access encoding.

Table 2-6 MVFR1_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	001

2.7 Media and VFP Feature Register 2

The MVFR2_EL1 characteristics are:

Purpose

Describes the features provided by the AArch32 Advanced SIMD and floating-point implementation.

Usage constraints

This register is accessible as follows:

EL0	EL1(NS)	EL1(S)	EL2	EL3 (SCR.NS = 1)	EL3(SCR.NS = 0)
-	RO	RO	RO	RO	RO

Configurations

MVFR2_EL1 is architecturally mapped to AArch32 register MVFR2. See *3.7 Media and VFP Feature Register 2* on page 3-40.

Attributes

MVFR2_EL1 is a 32-bit register.

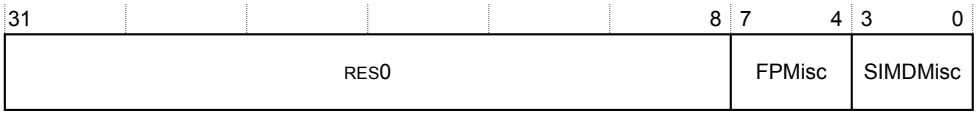

Figure 2-5 MVFR2_EL1 bit assignments

[31:8]

Reserved, RES0.

FPMisc, [7:4]

Indicates support for miscellaneous floating-point features.

`0x4` Supports:

- Floating-point selection.
- Floating-point Conversion to Integer with Directed Rounding modes.
- Floating-point Round to Integral Floating-point.
- Floating-point MaxNum and MinNum.

SIMDMisc, [3:0]

Indicates support for miscellaneous Advanced SIMD features.

`0x3` Supports:

- Floating-point Conversion to Integer with Directed Rounding modes.
- Floating-point Round to Integral Floating-point.
- Floating-point MaxNum and MinNum.

To access the MVFR2_EL1:

```
MRS <Xt>, MVFR2_EL1 ; Read MVFR2_EL1 into Xt
```

The following table shows the register access encoding.

Table 2-7 MVFR2_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0011	010

2.8 Floating-point Exception Control Register

The FPEXC32_EL2 characteristics are:

Purpose

Provides access to the AArch32 register FPEXC from AArch64 state only. Its value has no effect on execution in AArch64 state.

Usage constraints

This register is accessible as follows:

EL0	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	-	RW	RW	RW

Configurations

FPEXC32_EL2 is architecturally mapped to AArch32 register FPEXC. See *3.8 Floating-Point Exception Control register* on page 3-42.

Attributes

FPEXC32_EL2 is a 32-bit register.

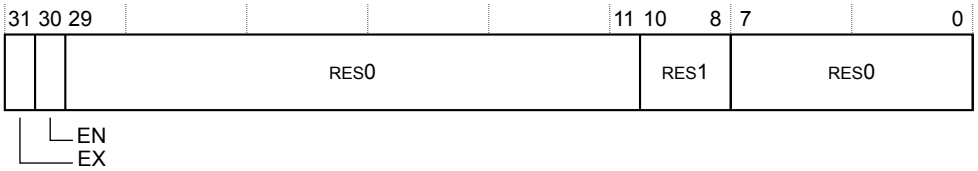

Figure 2-6 FPEXC32_EL2 bit assignments

EX, [31]

Exception bit.

RES0 The Cortex-A35 processor implementation does not generate asynchronous floating-point exceptions.

EN, [30]

Enable bit. A global enable for the Advanced SIMD and floating-point support:

0 The Advanced SIMD and floating-point support is disabled. This is the reset value.

1 The Advanced SIMD and floating-point support is enabled and operates normally.

This bit applies only to AArch32 execution, and only when EL1 is not AArch64.

[29:11]

Reserved, RES0.

[10:8]

Reserved, RES1.

[7:0]

Reserved, RES0.

To access the FPEXC32_EL2:

```
MRS <Xt>, FPEXC32_EL2 ; Read FPEXC32_EL2 into Xt
MSR FPEXC32_EL2, <Xt> ; Write Xt to FPEXC32_EL2
```

See also *2.1 Accessing the AArch64 feature identification registers* on page 2-14.

The following table shows the register access encoding.

Table 2-8 FPEXC32_EL2 access encoding

op0	op1	CRn	CRm	op2
11	100	0101	0011	000

Chapter 3
AArch32 Register Descriptions

This chapter describes the AArch32 registers for the Cortex-A35 processor Advanced SIMD and floating-point support.

It contains the following sections:

3.1 Accessing the AArch32 feature identification registers

Software can identify the Advanced SIMD and floating-point features using the feature identification registers in the AArch32 execution state.

You can access the feature identification registers in the AArch32 execution state using the `VMRS` instruction, for example:

```
VMRS <Rt>, FPSID ; Read FPSID into Rt
VMRS <Rt>, MVFR0 ; Read MVFR0 into Rt
VMRS <Rt>, MVFR1 ; Read MFFR1 into Rt
VMRS <Rt>, MVFR2 ; Read MVFR2 into Rt
```

Table 3-1 AArch32 Advanced SIMD and scalar floating-point feature identification registers

AArch32 name	Description
FPSID	See *3.3 Floating-Point System ID Register* on page 3-31.
MVFR0	See *3.5 Media and VFP Feature Register 0* on page 3-36.
MVFR1	See *3.6 Media and VFP Feature Register 1* on page 3-38.
MVFR2	See *3.7 Media and VFP Feature Register 2* on page 3-40.

3.2 AArch32 register summary

The processor has several Advanced SIMD and floating-point system registers in the AArch32 execution state. Each register has a specific purpose, usage constraints, configurations, and attributes.

The following table gives a summary of the Cortex-A35 processor Advanced SIMD and floating-point system registers in the AArch32 execution state.

Table 3-2 AArch32 Advanced SIMD and floating-point system registers

Name	Type	Reset	Description
FPSID	RO	`0x41034043`	See *3.3 Floating-Point System ID Register* on page 3-31.
FPSCR	RW	`0x00000000`	See *3.4 Floating-Point Status and Control Register* on page 3-33.
MVFR0	RO	`0x10110222`	See *3.5 Media and VFP Feature Register 0* on page 3-36.
MVFR1	RO	`0x12111111`	See *3.6 Media and VFP Feature Register 1* on page 3-38.
MVFR2	RW	`0x00000043`	See *3.7 Media and VFP Feature Register 2* on page 3-40.
FPEXC	RW	`0x00000700`	See *3.8 Floating-Point Exception Control register* on page 3-42.

Note

The *Floating-Point Instruction Registers*, FPINST and FPINST2 are not implemented, and any attempt to access them is UNPREDICTABLE.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for information on permitted accesses to the Advanced SIMD and floating-point system registers.

3.3 Floating-Point System ID Register

The FPSID characteristics are:

Purpose

Provides top-level information about the floating-point implementation.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Configurations

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, and HCPTR.{TCP10,TCP11}. For details of which field values permit access at specific exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* .

This register largely duplicates information held in the MIDR. Arm deprecates use of it.

Attributes

FPSID is a 32-bit register.

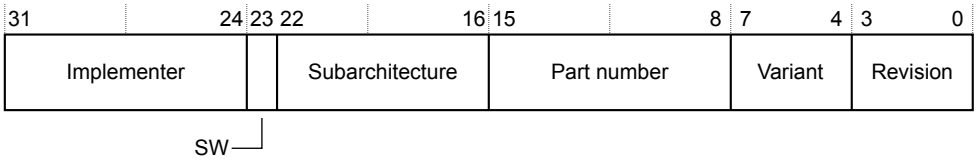

Figure 3-1 FPSID bit assignments

Implementer, [31:24]

Indicates the implementer:

`0x41` Arm Limited.

SW, [23]

Software bit. This bit indicates that a system provides only software emulation of the floating-point instructions:

`0` The system includes hardware support for floating-point operations.

Subarchitecture, [22:16]

Subarchitecture version number:

`0x03` VFPv3 architecture, or later, with no subarchitecture. The entire floating-point implementation is in hardware, and requires no software support code. The MVFR0, MVFR1 and MVFR2 registers indicate the VFP architecture version.

Part number, [15:8]

Indicates the part number for the floating-point implementation:

`0x40` v8-A profile

Variant, [7:4]

Indicates the variant number:

`0x4` Cortex-A35 processor.

Revision, [3:0]

Indicates the revision number for the floating-point implementation:

`0x3` r1p0.

To access the FPSID:

```
VMRS <Rt>, FPSID ; Read FPSID into Rt
```

3.4 Floating-Point Status and Control Register

The FPSCR characteristics are:

Purpose

Provides floating-point system status information and control.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
Config	RW	Config	RW	Config	Config	RW

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, HCPTR.{TCP10,TCP11} and FPEXC.EN. For details of which values of these fields allow access at which exception levels, see the *ARM® Architecture Reference Manual ARMv8, for ARMv8-A architecture profile*.

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

FPSCR is a 32-bit register.

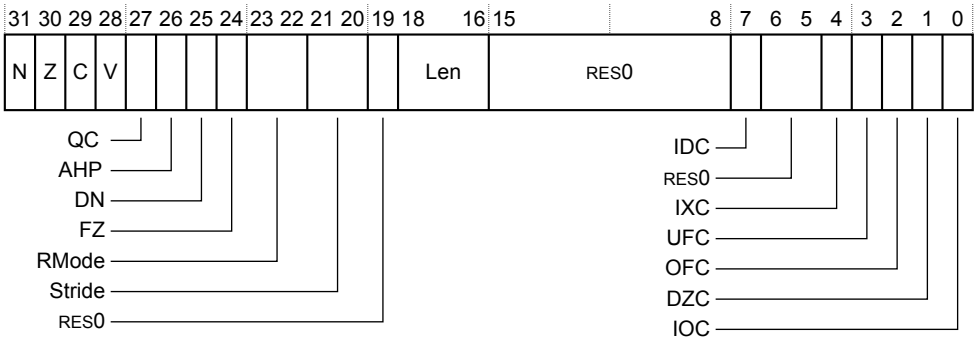

Figure 3-2 FPSCR bit assignments

N, [31]

Floating-point Negative condition code flag.

Set to 1 if a floating-point comparison operation produces a less than result.

Z, [30]

Floating-point Zero condition code flag.

Set to 1 if a floating-point comparison operation produces an equal result.

C, [29]

Floating-point Carry condition code flag.

Set to 1 if a floating-point comparison operation produces an equal, greater than, or unordered result.

V, [28]

Floating-point Overflow condition code flag.

Set to 1 if a floating-point comparison operation produces an unordered result.

QC, [27]

Cumulative saturation bit.

This bit is set to 1 to indicate that an Advanced SIMD integer operation has saturated after 0 was last written to this bit.

AHP, [26]

Alternative Half-Precision control bit:

`0` IEEE half-precision format selected. This is the reset value.

`1` Alternative half-precision format selected.

DN, [25]

Default NaN mode control bit:

`0` NaN operands propagate through to the output of a floating-point operation. This is the reset value.

`1` Any operation involving one or more NaNs returns the Default NaN.

The value of this bit only controls floating-point arithmetic. AArch32 Advanced SIMD arithmetic always uses the Default NaN setting, regardless of the value of the DN bit.

FZ, [24]

Flush-to-zero mode control bit:

`0` Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. This is the reset value.

`1` Flush-to-zero mode enabled.

The value of this bit only controls floating-point arithmetic. AArch32 Advanced SIMD arithmetic always uses the Flush-to-zero setting, regardless of the value of the FZ bit.

RMode, [23:22]

Rounding Mode control field:

`0b00` *Round to Nearest* (RN) mode. This is the reset value.

`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 almost all floating-point instructions. AArch32 Advanced SIMD arithmetic always uses the Round to Nearest setting, regardless of the value of the RMode bits.

Stride, [21:20]

RES0.

[19]

Reserved, RES0.

Len, [18:16]

RES0.

[15:8]

Reserved, RES0.

IDC, [7]

Input Denormal cumulative exception bit. This bit is set to 1 to indicate that the Input Denormal exception has occurred since 0 was last written to this bit.

[6:5]

Reserved, RES0.

IXC, [4]

Inexact cumulative exception bit. This bit is set to 1 to indicate that the Inexact exception has occurred since 0 was last written to this bit.

UFC, [3]

Underflow cumulative exception bit. This bit is set to 1 to indicate that the Underflow exception has occurred since 0 was last written to this bit.

OFC, [2]

Overflow cumulative exception bit. This bit is set to 1 to indicate that the Overflow exception has occurred since 0 was last written to this bit.

DZC, [1]

Division by Zero cumulative exception bit. This bit is set to 1 to indicate that the Division by Zero exception has occurred since 0 was last written to this bit.

IOC, [0]

Invalid Operation cumulative exception bit. This bit is set to 1 to indicate that the Invalid Operation exception has occurred since 0 was last written to this bit.

To access the FPSCR:

```
VMRS <Rt>, FPSCR ; Read FPSCR into RtVMSR FPSCR, <Rt> ; Write Rt to FPSCR
```

Note

The Cortex-A35 processor implementation does not support the deprecated VFP short vector feature. Attempts to execute the associated VFP data-processing instructions result in an Undefined Instruction exception.

3.5 Media and VFP Feature Register 0

The MVFR0 characteristics are:

Purpose

Describes the features provided by the AArch32 Advanced SIMD and floating-point implementation.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, HCPTR.{TCP10,TCP11}, and FPEXC.EN. For details of which values of these fields allow access at which exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* .

MVFR0 must be interpreted with MVFR1 and MVFR2. See *3.6 Media and VFP Feature Register 1* on page 3-38 and *3.7 Media and VFP Feature Register 2* on page 3-40.

Configurations

MVFR0 is architecturally mapped to AArch64 register MVFR0_EL1. See *2.5 Media and VFP Feature Register 0* on page 2-20.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

MVFR0 is a 32-bit register.

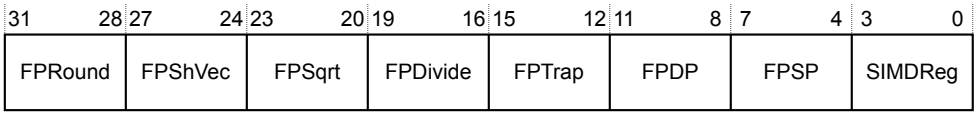

Figure 3-3 MVFR0 bit assignments

FPRound, [31:28]

Indicates the rounding modes supported by the floating-point hardware:

`0x1` All rounding modes supported.

FPShVec, [27:24]

Indicates the hardware support for floating-point short vectors:

`0x0` Not supported.

FPSqrt, [23:20]

Indicates the hardware support for floating-point square root operations:

`0x1` Supported.

FPDivide, [19:16]

Indicates the hardware support for floating-point divide operations:

`0x1` Supported.

FPTrap, [15:12]

Indicates whether the floating-point hardware implementation supports exception trapping:

`0x0` Not supported.

FPDP, [11:8]

Indicates the hardware support for floating-point double-precision operations:

`0x2` Supported, VFPv3 or greater.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

FPSP, [7:4]

Indicates the hardware support for floating-point single-precision operations:

`0x2` Supported, VFPv3 or greater.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

SIMDReg, [3:0]

Indicates support for the Advanced SIMD register bank:

`0x2` Supported, 32 x 64-bit registers supported.

See the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* for more information.

To access the MVFR0:

```
VMRS <Rt>, MVFR0 ; Read MVFR0 into Rt
```

3.6 Media and VFP Feature Register 1

The MVFR1 characteristics are:

Purpose

Describes the features provided by the AArch32 Advanced SIMD and floating-point implementation.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, HCPTR.{TCP10,TCP11}, and FPEXC.EN. For details of which values of these fields allow access at which exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* .

MVFR1 must be interpreted with MVFR0 and MVFR2. See *3.5 Media and VFP Feature Register 0* on page 3-36 and *3.7 Media and VFP Feature Register 2* on page 3-40

Configurations

MVFR1 is architecturally mapped to AArch64 register MVFR1_EL1. See *2.6 Media and VFP Feature Register 1* on page 2-22.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

MVFR1 is a 32-bit register.

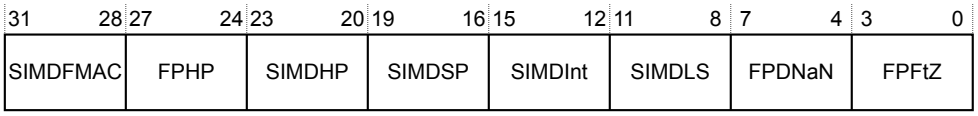

Figure 3-4 MVFR1 bit assignments

SIMDFMAC, [31:28]

Indicates whether the Advanced SIMD and floating-point unit supports fused multiply accumulate operations:

`0x1` Implemented.

FPHP, [27:24]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion instructions:

`0x2` Instructions to convert between half-precision and single-precision, and between half-precision and double-precision, implemented.

SIMDHP, [23:20]

Indicates whether the Advanced SIMD and floating-point unit supports half-precision floating-point conversion operations:

`0x1` Implemented.

SIMDSP, [19:16]

Indicates whether the Advanced SIMD and floating-point unit supports single-precision floating-point operations:

`0x1` Implemented.

SIMDInt, [15:12]

Indicates whether the Advanced SIMD and floating-point unit supports integer operations:

`0x1` Implemented.

SIMDLS, [11:8]

Indicates whether the Advanced SIMD and floating-point unit supports load/store instructions:

`0x1` Implemented.

FPDNaN, [7:4]

Indicates whether the floating-point hardware implementation supports only the Default NaN mode:

`0x1` Hardware supports propagation of NaN values.

FPFtZ, [3:0]

Indicates whether the floating-point hardware implementation supports only the Flush-to-Zero mode of operation:

`0x1` Hardware supports full denormalized number arithmetic.

To access the MVFR1:

```
VMRS <Rt>, MVFR1 ; Read MVFR1 into Rt
```

3.7 Media and VFP Feature Register 2

The MVFR2 characteristics are:

Purpose

Describes the features provided by the AArch32 Advanced SIMD and floating-point implementation.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RO	Config	Config	RO

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, HCPTR.{TCP10,TCP11}, and FPEXC.EN. For details of which values of these fields allow access at which exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* .

MVFR2 must be interpreted with MVFR0 and MVFR1. See *3.5 Media and VFP Feature Register 0* on page 3-36 and *3.6 Media and VFP Feature Register 1* on page 3-38

Configurations

MVFR2 is architecturally mapped to AArch64 register MVFR2_EL1. See *2.7 Media and VFP Feature Register 2* on page 2-24.

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

MVFR2 is a 32-bit register.

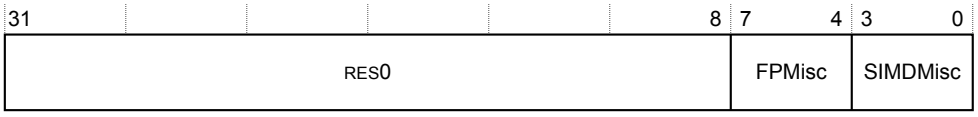

Figure 3-5 MVFR2 bit assignments

[31:8]

Reserved, RES0.

FPMisc, [7:4]

Indicates support for miscellaneous VFP features.

`0x4` Supports:

- Floating-point selection.
- Floating-point Conversion to Integer with Directed Rounding modes.
- Floating-point Round to Integral Floating-point.
- Floating-point MaxNum and MinNum.

SIMDMisc, [3:0]

Indicates support for miscellaneous Advanced SIMD features.

`0x3` Supports:

- Floating-point Conversion to Integer with Directed Rounding modes.
- Floating-point Round to Integral Floating-point.
- Floating-point MaxNum and MinNum.

To access the MVFR2:

```
VMRS <Rt>, MVFR2 ; Read MVFR2 into Rt
```

3.8 Floating-Point Exception Control register

The FPEXC characteristics are:

Purpose

Provides a global enable for the Advanced SIMD and floating-point support, and indicates how the state of this support is recorded.

Usage constraints

This register is accessible as follows:

EL0 (NS)	EL0 (S)	EL1 (NS)	EL1 (S)	EL2	EL3 (SCR.NS = 1)	EL3 (SCR.NS = 0)
-	-	Config	RW	Config	Config	RW

Access to this register depends on the values of CPACR.{cp10,cp11}, NSACR.{cp10,cp11}, and HCPTR.{TCP10,TCP11}. For details of which values of these fields allow access at which exception levels, see the *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile* .

Configurations

There is one copy of this register that is used in both Secure and Non-secure states.

Attributes

FPEXC is a 32-bit register.

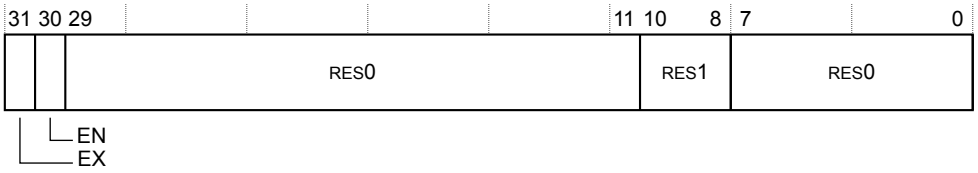

Figure 3-6 FPEXC bit assignments

EX, [31]

Exception bit. The Cortex-A35 processor implementation does not generate asynchronous floating-point exceptions, therefore this bit is RES0.

EN, [30]

Global enable for the Advanced SIMD and floating-point support:

`0` The Advanced SIMD and floating-point support is disabled. This is the reset value.

`1` The Advanced SIMD and floating-point support is enabled and operates normally.

It applies only to AArch32 executions, and only when EL1 is not AArch64.

[29:11]

Reserved, RES0.

[10:8]

Reserved, RES1.

[7:0]

Reserved, RES0.

To access the FPEXC register:

```
VMRS <Rt>, FPEXC ; Read FPEXC into Rt
```

Appendix A
Revisions

This appendix describes the technical changes between released issues of this book.

It contains the following section:

- *A.1 Revisions* on page Appx-A-44.

A.1 Revisions

This appendix describes the technical changes between released issues of this book.

Table A-1 Issue 0000-00

Change	Location	Affects
First release for r0p0	-	-

Table A-2 Issue 0001-00

Change	Location	Affects
First release for r0p1	On front page and in document history table.	r0p1
Product name udpated from the Mercury processor to the Cortex-A35 processor.	Everywhere the product name is used.	All versions

Table A-3 Issue 0002-00

Change	Location	Affects
First release for r0p2	On front page and in document history table.	r0p2

Table A-4 Issue 0100-00

Change	Location	Affects
First release for r1p0	On front page and in document history table.	r1p0