Cortex™-A7 Floating-Point Unit

Revision: r0p3

Technical Reference Manual

ARM DDI 0463D (ID052812)

Cortex-A7 Floating-Point Unit
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
13 September 2011	A	Non-Confidential	First release for r0p0
08 November 2011	B	Non-Confidential	First release for r0p1
11 January 2012	C	Non-Confidential	First release for r0p2
03 May 2012	D	Non-Confidential	First release for r0p3

Product Status

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

Web Address

`http://www.arm.com`

Contents
Cortex-A7 Floating-Point Unit Technical Reference Manual

Preface

This preface introduces the *Cortex-A7 Floating-Point Unit Technical Reference Manual*. It contains the following sections:

- *About this book* on page v
- *Feedback* on page viii.

About this book

This book is for the Cortex-A7 *Floating-Point Unit* (FPU) and describes the external functionality of the FPU.

Product revision status

The r*n*p*n* identifier indicates the revision status of the product described in this book, where:

r*n* Identifies the major revision of the product.

p*n* Identifies the minor revision or modification status of the product.

Intended audience

This book is written for system designers, system integrators, and programmers who are designing a *System-on-Chip* (SoC) device that uses the Cortex-A7 FPU.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for a high-level overview of the Cortex-A7 FPU and a description of its features.

Chapter 2 *Programmers Model*

Read this for a description of the Cortex-A7 FPU system registers.

Appendix A *Revisions*

Read this for a description of technical changes in this document.

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 *ARM Glossary*, `http://infocenter.arm.com/help/topic/com.arm.doc.aeg0014-/index.html`.

Conventions

Conventions that this book can use are described in:

- *Typographical*
- *Signals* on page vi.

Typographical

The typographical conventions are:

italic	Highlights important notes, introduces special terminology, denotes internal 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.

`monospace`	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 >	Enclose 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.

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.

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

Additional reading

This section lists publications by ARM and by third parties.

See Infocenter, http://infocenter.arm.com, for access to ARM documentation.

ARM publications

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

- *Cortex-A7 MPCore Technical Reference Manual* (ARM DDI 0464)
- *Cortex-A7 NEON Media Processing Engine Technical Reference Manual* (ARM DDI 0462)
- *Cortex-A7 MPCore Configuration and Sign-Off Guide (*ARM DII 0256)
- *Cortex-A7 MPCore Integration Manual* (ARM DIT 0017)
- *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406)
- *CoreSight™ ETM™ Cortex-A7 Technical Reference Manual* (ARM DDI 0468)
- *CoreSight ETM Cortex-A7 Configuration and Sign-Off Guide* (ARM DII 0261)
- *CoreSight Embedded Trace Macrocell v3.5 Architecture Specification* (ARM IHI 0014)
- *AMBA® AXI Protocol v1.0 Specification* (ARM IHI 0022)
- *ARM Generic Interrupt Controller Architecture Specification* (ARM IHI 0048)
- *RealView ICE User Guide* (ARM DUI 0155)

- *CoreSight Architecture Specification* (ARM IHI 0029)
- *CoreSight Technology System Design Guide* (ARM DGI 0012).

Other publications

This section lists relevant documents published by third parties:

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

Feedback

ARM welcomes feedback on this product and its documentation.

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
- the number, ARM DDI 0463D
- the page numbers to which your comments apply
- a concise explanation of your comments.

ARM also welcomes general suggestions for additions and improvements.

Chapter 1
Introduction

This chapter introduces the Cortex-A7 FPU. It contains the following sections:

1.1 About the Cortex-A7 FPU

The Cortex-A7 FPU is a VFPv4-D16 implementation of the ARMv7 floating-point architecture. It provides low-cost high performance floating-point computation. The Cortex-A7 FPU supports all addressing modes and operations described in the *ARM Architecture Reference Manual*.

The Cortex-A7 FPU features are:

- support for single-precision and double-precision floating-point formats
- support for conversion between half-precision and single-precision
- support for *Fused Multiply ACcumulate* (FMAC) operations
- normalized and denormalized data are all handled in hardware
- trapless operation enabling fast execution.

The Cortex-A7 FPU hardware supports single and double-precision add, subtract, multiply, divide, multiply and accumulate, fused multiply accumulate, and square root operations as described in the ARM VFPv4 architecture. It provides conversions between 16-bit, 32-bit, and 64-bit floating-point formats and ARM integer word formats, with special operations to perform conversions in round-towards-zero mode for high-level language support.

ARMv7 deprecates the use of VFP vector mode. The Cortex-A7 FPU hardware does not support VFP vector operations. The Cortex-A7 FPU provides high speed VFP operation without support code. However, if an application requires VFP vector operation, then it must use support code. See the *ARM Architecture Reference Manual* for information on VFP vector operation support.

Note

This manual gives information specific to the Cortex-A7 FPU implementation of the ARM VFPv4 extension. See the *ARM Architecture Reference Manual* for full instruction set and usage details.

1.2 Applications

The Cortex-A7 FPU provides floating-point computation suitable for a wide spectrum of applications such as:

- personal digital assistants and smartphones for graphics, voice compression and decompression, user interfaces, Java interpretation, and *Just In Time* (JIT) compilation
- games machines for three-dimensional graphics and digital audio
- printers and *MultiFunction Peripheral* (MFP) controllers for high-definition color rendering
- set-top boxes for digital audio and digital video, and three-dimensional user interfaces
- automotive applications for engine management and power train computations.

1.3 Product revisions

This section describes the differences in functionality between product revisions:

r0p0-r0p1 Functional changes are:

- ID register value changed to reflect product revision status:

  FPSID Register `0x41023071`

  ——— **Note** ———

  Product revision updated to maintain consistency with the main Cortex-A7 MPCore product.

  ————————

r0p1-r0p2 Functional changes are:

- ID register value changed to reflect product revision status:

  FPSID Register `0x41023072`

  ——— **Note** ———

  Product revision updated to maintain consistency with the main Cortex-A7 MPCore product.

  ————————

r0p2-r0p3 Functional changes are:

- ID register value changed to reflect product revision status:

  FPSID Register `0x41023073`

  ——— **Note** ———

  Product revision updated to maintain consistency with the main Cortex-A7 MPCore product.

  ————————

Chapter 2
Programmers Model

This chapter describes implementation-specific features of the Cortex-A7 FPU that are useful to programmers. It contains the following sections:

2.1 About the programmers model

This section introduces the Cortex-A7 FPU implementation of the VFPv4 floating-point architecture, VFPv4-D16, with version 2 of the Common VFP subarchitecture. In this implementation:

- All scalar operations are implemented entirely in hardware, with support for all combinations of rounding modes, flush-to-zero, and default NaN modes.
- Vector operations are not supported. Any attempt to execute a vector operation results in an Undefined Instruction exception with the FPEXC.DEX bit set to 1.
- The Cortex-A7 FPU never generates an asynchronous VFP exception.

In addition it provides information on initializing the Cortex-A7 FPU ready for application code execution.

2.1.1 VFP feature identification registers

The Cortex-A7 FPU implements the ARMv7 VFPv4 extension.

Software can identify this extension and the features it provides, using the feature identification registers. These registers are in extension is in the coprocessor space for coprocessors CP10 and CP11. You can access the registers using the `VMRS` and `VMSR` instructions, for example:

```
VMRS <Rd>, FPSID ; Read Floating-Point System ID Register
VMRS <Rd>, MVFR1 ; Read Media and VFP Feature Register 1
VMSR FPSCR, <Rt> ; Write Floating-Point System Control Register
```

See *VFP register access* on page 2-4 for a description of the registers.

2.1.2 Enabling VFP support

From reset, the Cortex-A7 FPU is disabled. Any attempt to execute a VFP instruction results in an Undefined Instruction exception being taken. To enable software to access VFP features ensure that:

- Access to CP10 and CP11 is enabled for the appropriate privilege level. See *VFP register access* on page 2-4.
- If Non-secure access to the VFP features is required, the access flags for CP10 and CP11 in the NSACR must be set to 1. See *VFP register access* on page 2-4.

In addition, software must set the FPEXC.EN bit to 1 to enable most VFP operations. See *Floating-Point Exception Register* on page 2-10.

When VFP operation is disabled because FPEXC.EN is 0, all VFP instructions are treated as Undefined instructions except for execution of the following in privileged modes:

- a `VMSR` to the FPEXC or FPSID register
- a `VMRS` from the FPEXC, FPSID, MVFR0, or MVFR1 registers.

To use the Cortex-A7 FPU in Secure state only

To use the Cortex-A7 FPU in Secure state only, define the CPACR and *Floating-Point Exception* (FPEXC) registers to enable the Cortex-A7 FPU:

1. Set the CPACR for access to CP10 and CP11:

   ```
   LDR r0, =(0xF << 20)
   MCR p15, 0, r0, c1, c0, 2
   ```

2. Set the FPEXC EN bit to enable the Cortex-A7 FPU:

```
MOV r3, #0x40000000
VMSR FPEXC, r3
```

At this point the Cortex-A7 processor can execute VFP instructions.

To use the Cortex-A7 FPU in Secure state and Non-secure state

To use the Cortex-A7 FPU in Secure state and Non-secure state, first define the NSACR and then define the CPACR and FPEXC registers to enable the Cortex-A7 FPU.

1. Set bits [11:10] of the NSACR for access to CP10 and CP11 from both Secure and Non-secure states:

```
MRC p15, 0, r0, c1, c1, 2
ORR r0, r0, #2_11<<10 ; enable fpu
MCR p15, 0, r0, c1, c1, 2
```

2. Set the CPACR for access to CP10 and CP11:

```
LDR r0, =(0xF << 20)
MCR p15, 0, r0, c1, c0, 2
```

3. Set the FPEXC EN bit to enable the Cortex-A7 FPU:

```
MOV r3, #0x40000000
VMSR FPEXC, r3
```

At this point the Cortex-A7 processor can execute VFP instructions.

Note

Operation is UNPREDICTABLE if you configure the *Coprocessor Access Control Register* (CPACR) such that CP10 and CP11 do not have identical access permissions.

2.2 VFP register access

Table 2-1 shows the system control coprocessor registers, accessed through CP15, that determine access to VFP registers, where:

- CRn is the register number within CP15
- Op1 is the Opcode_1 value for the register
- CRm is the operational register
- Op2 is the Opcode_2 value for the register.

Table 2-1 Coprocessor Access Control registers

CRn	Op1	CRm	Op2	Name	Description
c1	0	c0	2	CPACR	See the *Cortex-A7 MPCore Technical Reference Manual*
c1	0	c1	2	NSACR	See the *Cortex-A7 MPCore Technical Reference Manual*

2.3 Register summary

Table 2-2 shows the Cortex-A7 FPU system registers. All Cortex-A7 FPU system registers are 32-bit wide. Reserved register addresses are UNPREDICTABLE.

Table 2-2 Cortex-A7 FPU system registers

Name	Type	Reset	Description
FPSID	RO	`0x41023073`	*Floating-Point System ID Register* on page 2-6
FPSCR	RW	`0x00000000`	*Floating-Point Status and Control Register* on page 2-6
MVFR0	RO	`0x10110221`	*Media and VFP Feature Register 0* on page 2-8
MVFR1	RO	`0x11000011`	*Media and VFP Feature Register 1* on page 2-9
FPEXC	RW	`0x00000000`	*Floating-Point Exception Register* on page 2-10

Note

The FPINST and FPINST2 registers are not implemented, and any attempt to access them is UNPREDICTABLE.

2.3.1 Processor modes for accessing the Cortex-A7 FPU system registers

Table 2-3 shows the processor modes for accessing the Cortex-A7 FPU system registers.

Table 2-3 Accessing Cortex-A7 FPU system registers

Register	Privileged access		User access	
	FPEXC EN=0	FPEXC EN=1	FPEXC EN=0	FPEXC EN=1
FPSID	Permitted	Permitted	Not permitted	Not permitted
FPSCR	Not permitted	Permitted	Not permitted	Permitted
MVFR0, MVFR1	Permitted	Permitted	Not permitted	Not permitted
FPEXC	Permitted	Permitted	Not permitted	Not permitted

2.4 Register descriptions

This section describes the Cortex-A7 FPU system registers. Table 2-2 on page 2-5 provides cross references to individual registers.

2.4.1 Floating-Point System ID Register

The FPSID characteristics are:

Purpose Provides information about the VFP implementation.

Usage constraints This register is:

- Only accessible in the Non-secure state if the CP10 and CP11 bits in the NSACR are set to 1, see *VFP register access* on page 2-4.
- Only accessible in privileged modes, and only if access to coprocessors CP10 and CP11 is enabled in the CPACR and FPEXC.EN is set to 1, see *VFP register access* on page 2-4.

Configurations Available in all configurations.

Attributes See the register summary in Table 2-2 on page 2-5.

Figure 2-1 shows the FPSID bit assignments.

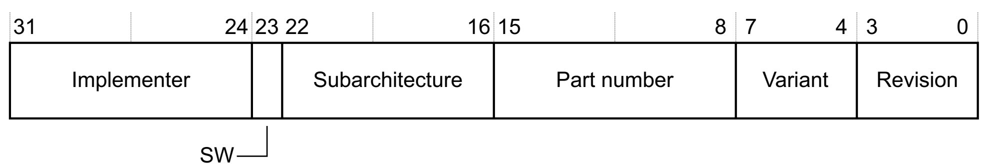

Figure 2-1 FPSID bit assignments

Table 2-4 shows the FPSID bit assignments.

Table 2-4 FPSID bit assignments

Bits	Name	Function
[31:24]	Implementer	Denotes ARM. Value is `0x41`.
[23]	SW	Hardware implementation with no software emulation. Value is `0x0`.
[22:16]	Subarchitecture	VFPv3 or greater with v2 subarchitecture. Value is `0x2`.
[15:8]	Part number	Cortex-A. Value is `0x30`.
[7:4]	Variant	Cortex-A7. Value is `0x7`.
[3:0]	Revision	Revision. Value is `0x3`.

You can access the FPSID with the following `VMRS` instruction:

```
VMRS <Rd>, FPSID ; Read Floating-Point System ID register
```

2.4.2 Floating-Point Status and Control Register

The FPSCR characteristics are:

Purpose Provides user-level control of the Cortex-A7 FPU.

Usage constraints This register is:

- Only accessible in the Non-secure state if the CP10 and CP11 bits in the NSACR are set to 1, see *VFP register access* on page 2-4.
- Accessible in all modes depending on the setting of bits [23:20] of the CPACR and FPEXC.EN, see *VFP register access* on page 2-4.

Configurations Available in all configurations.

Attributes See the register summary in Table 2-2 on page 2-5.

Figure 2-2 shows the FPSCR bit assignments.

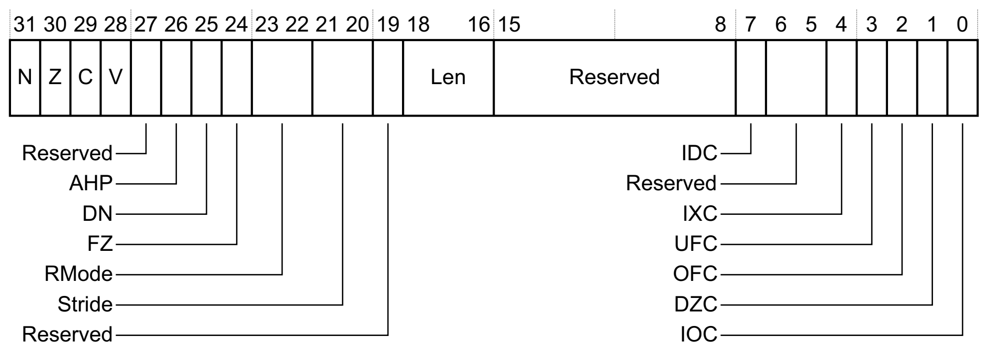

Figure 2-2 FPSCR bit assignments

Table 2-5 shows the FPSCR bit assignments.

Table 2-5 FPSCR bit assignments

Bits	Name	Function
[31]	N	Set to 1 if a comparison operation produces a less than result.
[30]	Z	Set to 1 if a comparison operation produces an equal result.
[29]	C	Set to 1 if a comparison operation produces an equal, greater than, or unordered result.
[28]	V	Set to 1 if a comparison operation produces an unordered result.
[27]	Reserved	UNK/SBZP.
[26]	AHP	Alternative Half-Precision control bit: **0** IEEE half-precision format selected. **1** Alternative half-precision.
[25]	DN	Default NaN mode control bit: **0** NaN operands propagate through to the output of a floating-point operation. **1** Any operation involving one or more NaNs returns the Default NaN.
[24]	FZ	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. **1** Flush-to-zero mode enabled.

Table 2-5 FPSCR bit assignments (continued)

Bits	Name	Function
[23:22]	RMode	Rounding Mode control field: `b00` *Round to Nearest* (RN) mode. `b01` *Round towards Plus infinity* (RP) mode. `b10` *Round towards Minus infinity* (RM) mode. `b11` *Round towards Zero* (RZ) mode.
[21:20]	Stride	Stride control used for backwards compatibility with short vector values. See the *ARM Architecture Reference Manual*.
[19]	Reserved	UNK/SBZP.
[18:16]	Len	Vector length, used for backwards compatibility with short vector values. See the *ARM Architecture Reference Manual*.
[15:8]	Reserved	UNK/SBZP.
[7]	IDC	Input Denormal cumulative exception flag.
[6:5]	Reserved	UNK/SBZP.
[4]	IXC	Inexact cumulative exception flag.
[3]	UFC	Underflow cumulative exception flag.
[2]	OFC	Overflow cumulative exception flag.
[1]	DZC	Division by Zero cumulative exception flag.
[0]	IOC	Invalid Operation cumulative exception flag.

You can access the FPSCR with the following `VMSR` instructions:

```
VMRS <Rd>, FPSCR ; Read Floating-Point Status and Control Register
VMSR FPSCR, <Rt> ; Write Floating-Point Status and Control Register
```

2.4.3 Media and VFP Feature Register 0

The MVFR0 characteristics are:

Purpose Together with MVFR1, describes the features that the Cortex-A7 FPU provides.

Usage constraints This register is:

- Only accessible in the Non-secure state if the CP10 and CP11 bits in the NSACR are set to 1, see *VFP register access* on page 2-4.
- Only accessible in privileged modes, and only if access to coprocessors CP10 and CP11 is enabled in the CPACR and FPEXC.EN is set to 1, see *VFP register access* on page 2-4.

Configurations Available in all configurations.

Attributes See the register summary in Table 2-2 on page 2-5.

Figure 2-3 on page 2-9 shows the MVFR0 bit assignments.

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
VFP rounding modes	Short vectors	Square root	Divide	VFP exception trapping	Double-precision	Single-precision	A_SIMD registers

Figure 2-3 MVFR0 bit assignments

Table 2-6 shows the MVFR0 bit assignments.

Table 2-6 MVFR0 bit assignments

Bits	Name	Function
[31:28]	VFP rounding modes	All VFP rounding modes supported. Value is `0x1`.
[27:24]	Short vectors	VFP short vectors not supported. Value is `0x0`.
[23:20]	Square root	VFP square root operation supported. Value is `0x1`.
[19:16]	Divide	VFP divide operation supported. Value is `0x1`.
[15:12]	VFP exception trapping	VFP exception trapping not supported. Value is `0x0`.
[11:8]	Double-precision	Double-precision operations supported. Value is `0x2`.
[7:4]	Single-precision	Single-precision operations supported. Value is `0x2`.
[3:0]	A_SIMD registers	Sixteen 64-bit registers supported. Value is `0x1`.

You can access the MVFR0 with the following `VMSR` instruction:

```
VMRS <Rd>, MVFR0 ; Read Media and VFP Feature Register 0
```

2.4.4 Media and VFP Feature Register 1

The MVFR1 characteristics are:

Purpose Together with MVFR0, describes the features that the Cortex-A7 FPU provides.

Usage constraints This register is:

- Only accessible in the Non-secure state if the CP10 and CP11 bits in the NSACR are set to 1, see *VFP register access* on page 2-4.
- Only accessible in privileged modes, and only if access to coprocessors CP10 and CP11 is enabled in the CPACR and FPEXC.EN is set to 1, see *VFP register access* on page 2-4.

Configurations Available in all configurations.

Attributes See the register summary in Table 2-2 on page 2-5.

Figure 2-4 shows the MVFR1 bit assignments.

31–28	27–24	23–20	19–16	15–12	11–8	7–4	3–0
A_SIMD FMAC	VFP HPFP	A_SIMD HPFP	A_SIMD SPFP	A_SIMD integer	A_SIMD load/store	D_NaN mode	FtZ mode

Figure 2-4 MVFR1 bit assignments

Table 2-7 shows the MVFR1 bit assignments.

Table 2-7 MVFR1 bit assignments

Bits	Name	Function
[31:28]	A_SIMD FMAC	Fused Multiply Accumulate supported. Value is `0x1`.
[27:24]	VFP HPFP	VFP half-precision operations supported. Value is `0x1`.
[23:20]	A_SIMD HPFP	Advanced SIMD half-precision operations not supported. Value is `0x0`.
[19:16]	A_SIMD SPFP	Advanced SIMD single-precision operations not supported. Value is `0x0`.
[15:12]	A_SIMD integer	Advanced SIMD integer operations not supported. Value is `0x0`.
[11:8]	A_SIMD load/store	Advanced SIMD load/store instructions not supported. Value is `0x0`.
[7:4]	D_NaN mode	Propagation of NaN values supported for VFP. Value is `0x1`.
[3:0]	FtZ mode	Full denormal arithmetic operations supported for VFP. Value is `0x1`.

You can access the MVFR1 with the following `VMSR` instruction:

```
VMRS <Rd>, MVFR1 ; Read Media and VFP Feature Register 1
```

2.4.5 Floating-Point Exception Register

The FPEXC characteristics are:

Purpose Provides global enable control of the VFP extension.

Usage constraints This register is:

- Only accessible in the Non-secure state if the CP10 and CP11 bits in the NSACR are set to 1, see *VFP register access* on page 2-4.
- Only accessible in privileged modes, and only if access to coprocessors CP10 and CP11 is enabled in the CPACR, see *VFP register access* on page 2-4.

Configurations Available in all configurations.

Attributes See the register summary in Table 2-2 on page 2-5.

Figure 2-5 shows the FPEXC bit assignments.

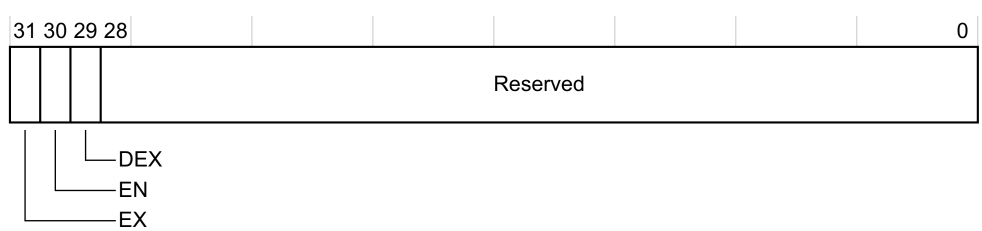

Figure 2-5 FPEXC bit assignments

Table 2-8 shows the FPEXC bit assignments.

Table 2-8 FPEXC bit assignments

Bits	Name	Function
[31]	EX	The Cortex-A7 FPU does not generate asynchronous VFP exceptions, therefore this bit is RAZ/WI.
[30]	EN	Cortex-A7 FPU enable bit: **0** FPU disabled. **1** FPU enabled. The EN bit is cleared to 0 at reset.
[29]	DEX	Defined synchronous instruction exceptional flag. The Cortex-A7 FPU sets this bit when generating a synchronous bounce because of an attempt to execute a vector operation. All other Undefined Instruction exceptions clear this bit to zero. See the *ARM Architecture Reference Manual* for more information.
[28:0]	Reserved	RAZ/WI.

You can access the FPEXC with the following `VMSR` instructions:

```
VMRS <Rd>, FPEXC ; Read Floating-Point Exception Register
VMSR FPEXC, <Rt> ; Write Floating-Point Exception Register
```

Appendix A
Revisions

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

Table A-1 Issue A

Change	Location	Affects
No changes, first release	-	-

Table A-2 Differences between issue A and issue B

Change	Location	Affects
Updated reset value of the FPSID Register	Table 2-2 on page 2-5	r0p1
Updated bits[3:0] of the FPSID Register	Table 2-4 on page 2-6	r0p1
Clarified the value for MVFR0.A_SIMD register field	Table 2-6 on page 2-9	All

Table A-3 Differences between issue B and issue C

Change	Location	Affects
Updated reset value of the FPSID Register	Table 2-2 on page 2-5	r0p2
Updated bits[3:0] of the FPSID Register	Table 2-4 on page 2-6	r0p2

Table A-4 Differences between issue C and issue D

Change	Location	Affects
Updated reset value of the FPSID Register	Table 2-2 on page 2-5	r0p3
Updated bits[3:0] of the FPSID Register	Table 2-4 on page 2-6	r0p3