Cortex™-A5 Floating-Point Unit

Revision: r0p0

Technical Reference Manual

ARM DDI 0449A (ID012010)

Cortex-A5 Floating-Point Unit
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
18 December 2009	A	Non-Confidential	First release for r0p0

Proprietary Notice

Words and logos marked with ® or ™ are registered trademarks or trademarks of ARM® in the EU and other countries, except as otherwise stated below in this proprietary notice. Other brands and names mentioned herein may be the trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Where the term ARM is used it means "ARM or any of its subsidiaries as appropriate".

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

Cortex-A5 Floating-Point Unit Technical Reference Manual

List of Tables
Cortex-A5 Floating-Point Unit Technical Reference Manual

List of Figures
Cortex-A5 Floating-Point Unit Technical Reference Manual

Preface

This preface introduces the *Cortex-A5 Floating-Point Unit Technical Reference Manual*. It contains the following sections:

- *About this book* on page vii
- *Feedback* on page x.

About this book

This book is for the Cortex-A5 *Floating-Point Unit* (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 verification engineers who are designing a *System-on-Chip* (SoC) device that uses the FPU. The book describes the external functionality of the FPU.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for a high-level view of the Cortex-A5 FPU and a description of its features.

Chapter 2 *Programmers Model*

Read this for a description of the Cortex-A5 FPU system registers.

Appendix A *Revisions*

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

Glossary Read this for definitions of terms used in this book.

Conventions

Conventions that this book can use are described in:

- *Typographical*
- *Signals* on page viii.

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>`

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-A5 Technical Reference Manual* (ARM DDI 0433)
- *Cortex-A5 MPCore Technical Reference Manual* (ARM DDI 0434)
- *Cortex-A5 Supplementary Datasheet* (ARM DDI 0448)
- *Cortex-A5 NEON Media Processing Engine Technical Reference Manual* (ARM DDI 0450)
- *Cortex-A5 Configuration and Sign-Off Guide (*ARM DII 0210)
- *Cortex-A5 Integration Manual* (ARM DIT 0001)
- *ARM Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406)
- *CoreSight™ ETM™-A5 Technical Reference Manual* (ARM DDI 0435)
- *CoreSight ETM-A5 Configuration and Sign-Off Guide* (ARM DII 0212)
- *CoreSight ETM-A5 Integration Manual* (ARM DIT 0002)
- *CoreSight Design Kit for Cortex-A5 Integration Manual* (ARM DIT 0003)
- *CoreSight Embedded Trace Macrocell v3.5 Architecture Specification* (ARM IHI 0014)
- *PrimeCell Level 2 Cache Controller (PL310) Technical Reference Manual* (ARM DDI 0246)
- *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)

- *Intelligent Energy Controller Technical Overview* (ARM DTO 0005)
- *CoreSight Architecture Specification* (ARM IHI 0029)
- *CoreSight Technology System Design Guide* (ARM DGI 0012)
- *Cortex-A5 FPU Release Note* (AT551-DC-06002).

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 0449A
- 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-A5 FPU. It contains the following sections:

- *About the Cortex-A5 FPU* on page 1-2
- *Applications* on page 1-3
- *Product revisions* on page 1-4.

1.1 About the Cortex-A5 FPU

The Cortex-A5 FPU is a VFPv4-D16 implementation of the ARMv7 floating-point architecture. It provides low-cost high performance floating-point computation. The FPU supports all addressing modes and operations described in the *ARM Architecture Reference Manual*.

The 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* (FMA) operations
- normalized and denormalized data are all handled in hardware
- trapless operation enabling fast execution.

The Cortex-A5 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-A5 FPU hardware does not support VFP vector operations. The Cortex-A5 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-A5 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-A5 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	First release.

Chapter 2
Programmers Model

This chapter describes implementation-specific features of the Cortex-A5 FPU that are useful to programmers. It contains the following sections:

2.1 About the programmers model

This section introduces the Cortex-A5 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 a synchronous bounce, with the FPEXC.DEX bit set to 1.
- The Cortex-A5 FPU never generates an asynchronous VFP exception.

In addition it provides information on initializing the Cortex-A5 FPU ready for application code execution.

Table 2-2 on page 2-5 describes the following access types:

RW	Read and write.
RO	Read only.

2.1.1 VFP feature identification registers

The Cortex-A5 FPU implements the ARMv7 VFP extension.

Software can identify this extension and the features it provides, using the feature identification registers. This 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.

In addition there are coprocessor access control registers. See *VFP register access* on page 2-4.

2.1.2 Enabling VFP support

From reset, the VFP extension 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 FPU in Secure state only

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

1. Set the CPACR for access to CP10 and CP11 (the FPU coprocessors):

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

2. Set the FPEXC EN bit to enable the FPU:

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

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

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

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

At this point the Cortex-A5 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-A5 Technical Reference Manual*
c1	0	c1	2	NSACR	See the *Cortex-A5 Technical Reference Manual*

2.3 Register summary

Table 2-2 shows the Cortex-A5 FPU system registers. All FPU system registers are 32-bit wide. Reserved register addresses are UNPREDICTABLE.

Table 2-2 Cortex-A5 FPU system registers

Name	Type	Reset	Description
FPSID	RO	`0x41023050`	*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 FPU system registers

Table 2-3 shows the processor modes for accessing the Cortex-A5 FPU system registers.

Table 2-3 Accessing Cortex-A5 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 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 Register 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, 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 Register bit assignments.

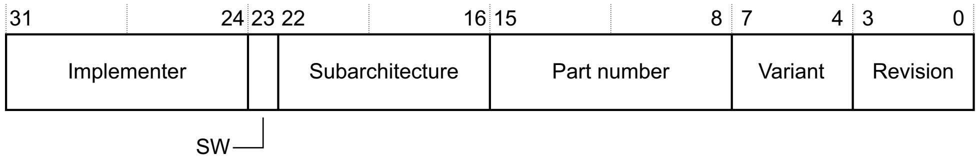

Figure 2-1 FPSID Register bit assignments

Table 2-4 shows the FPSID Register bit assignments.

Table 2-4 FPSID Register bit assignments

Bits	Name	Function
[31:24]	Implementer	Denotes ARM. Value is `0x41`.
[23]	SW	Hardware implementation with no software emulation. Value is `0`.
[22:16]	Subarchitecture	VFPv3 or greater with v2 subarchitecture. Value is `2`.
[15:8]	Part number	Cortex-A. Value is `0x30`.
[7:4]	Variant	Cortex-A5. Value is `5`.
[3:0]	Revision	Revision. Value is `0`.

You can access the FPSID Register 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 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.

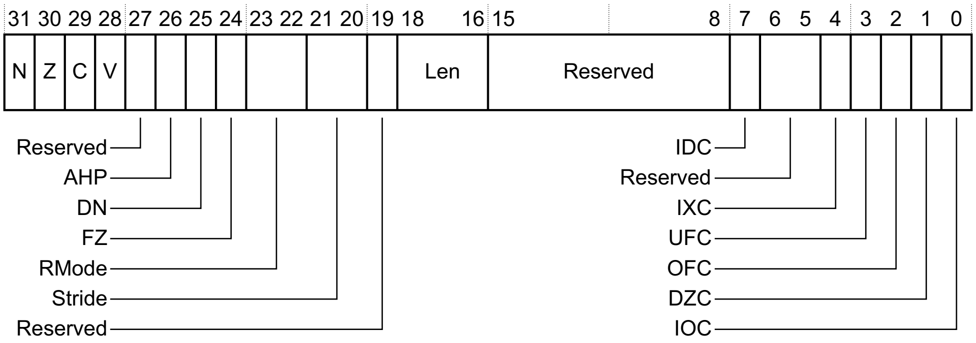

Figure 2-2 FPSCR bit assignments

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: b0 = IEEE half-precision format selected b1 = Alternative half-precision.
[25]	DN	Default NaN mode control bit: b0 = NaN operands propagate through to the output of a floating-point operation. b1 = Any operation involving one or more NaNs returns the Default NaN.
[24]	FZ	Flush-to-zero mode control bit: b0 = Flush-to-zero mode disabled. Behavior of the floating-point system is fully compliant with the IEEE 754 standard. b1 = 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 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, 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.

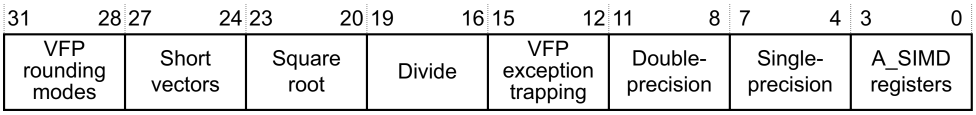

Figure 2-3 MVFR0 bit assignments

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
[27:24]	Short vectors	VFP short vectors not supported
[23:20]	Square root	VFP square root operation supported
[19:16]	Divide	VFP divide operation supported
[15:12]	VFP exception trapping	VFP exception trapping not supported
[11:8]	Double-precision	Double-precision operations supported
[7:4]	Single-precision	Single-precision operations supported
[3:0]	A_SIMD registers	Sixteen 64-bit registers supported

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 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, 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.

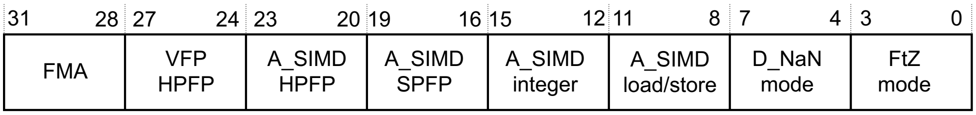

Figure 2-4 MVFR1 bit assignments

Table 2-7 shows the MVFR1 bit assignments.

Table 2-7 MVFR1 bit assignments

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

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 Register 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 Register bit assignments.

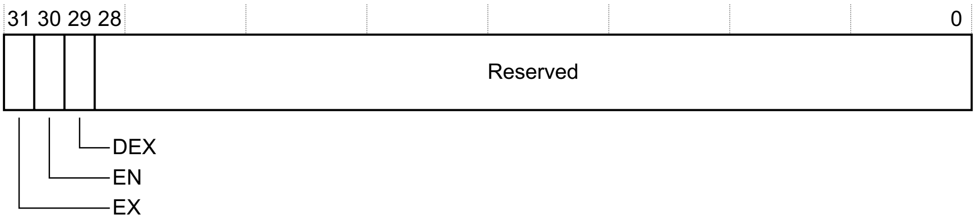

Figure 2-5 FPEXC Register bit assignments

Table 2-8 shows the FPEXC Register bit assignments.

Table 2-8 FPEXC Register bit assignments

Bits	Name	Function
[31]	EX	The Cortex-A5 FPU does not generate asynchronous VFP exceptions, therefore this bit is RAZ/WI.
[30]	EN	FPU enable bit: b0 = FPU disabled. b1 = FPU enabled. The EN bit is cleared to 0 at reset.
[29]	DEX	Defined synchronous instruction exceptional flag. The Cortex-A5 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 Register with the following `VMSR` instructions:

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

Appendix A
Revisions

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

Table A-1 Issue A

Change	Location
No changes, first release	-

Glossary

This glossary describes some of the terms used in ARM manuals. Where terms can have several meanings, the meaning presented here is intended.

Abort	A mechanism that indicates to a core that the value associated with a memory access is invalid. An abort can be caused by the external or internal memory system as a result of attempting to access invalid instruction or data memory. An abort is classified as either a Prefetch or Data Abort, and an internal or External Abort. *See also* Data Abort, External Abort and Prefetch Abort.
Addressing modes	A mechanism, shared by many different instructions, for generating values used by the instructions. For four of the ARM addressing modes, the values generated are memory addresses (which is the traditional role of an addressing mode). A fifth addressing mode generates values to be used as operands by data-processing instructions.
Aligned	A data item stored at an address that is divisible by the number of bytes that defines the data size is said to be aligned. Aligned words and halfwords have addresses that are divisible by four and two respectively. The terms word-aligned and halfword-aligned therefore stipulate addresses that are divisible by four and two respectively.
ARM instruction	A word that specifies an operation for an ARM processor to perform. ARM instructions must be word-aligned.
ARM state	A processor that is executing ARM (32-bit) word-aligned instructions is operating in ARM state.
Bounce	The FPU coprocessor bounces an instruction when it fails to signal the acceptance of a valid FPU instruction to the ARM processor. This action initiates Undefined instruction processing by the ARM processor. The FPU support code is called to complete the instruction that was found to be exceptional or unsupported by the FPU coprocessor. *See also* Trigger instruction, Potentially exceptional instruction, and Exceptional state.

Byte	An 8-bit data item.
Condition field	A four-bit field in an instruction that specifies a condition under which the instruction can execute.
Conditional execution	If the condition code flags indicate that the corresponding condition is true when the instruction starts executing, it executes normally. Otherwise, the instruction does nothing.
Control bits	The bottom eight bits of a Program Status Register. The control bits change when an exception arises and can be altered by software only when the processor is in a privileged mode.
Coprocessor	A processor that supplements the main processor. It carries out additional functions that the main processor cannot perform. Usually used for floating-point math calculations, signal processing, or memory management.
Core	A core is that part of a processor that contains the ALU, the datapath, the general-purpose registers, the Program Counter, and the instruction decode and control circuitry.
CPSR	*See* Current Program Status Register
Current Program Status Register (CPSR)	The register that holds the current operating processor status.
Default NaN mode	A mode in which all operations that result in a NaN return the default NaN, regardless of the cause of the NaN result. This mode is compliant with the IEEE 754 standard but implies that all information contained in any input NaNs to an operation is lost.
Denormalized value	*See* Subnormal value.
Disabled exception	An exception is disabled when its exception enable bit in the FPCSR is not set. For these exceptions, the IEEE 754 standard defines the result to be returned. An operation that generates an exception condition can bounce to the support code to produce the result defined by the IEEE 754 standard. The exception is not reported to the user trap handler.
DNM	*See* Do Not Modify.
Do Not Modify (DNM)	In Do Not Modify fields, the value must not be altered by software. DNM fields read as Unpredictable values, and must only be written with the same value read from the same field on the same processor. DNM fields are sometimes followed by RAZ or RAO in parentheses to show which way the bits must read for future compatibility, but programmers must not rely on this behavior.
Double-precision value	Consists of two 32-bit words that must appear consecutively in memory and must both be word-aligned, and that is interpreted as a basic double-precision floating-point number according to the IEEE 754-1985 standard.
Doubleword	A 64-bit data item. The contents are taken as being an unsigned integer unless otherwise stated.
Enabled exception	An exception is enabled when its exception enable bit in the FPCSR is set. When an enabled exception occurs, a trap to the user handler is taken. An operation that generates an exception condition might bounce to the support code to produce the result defined by the IEEE 754 standard. The exception is then reported to the user trap handler.
Exception	A fault or error event that is considered serious enough to require that program execution is interrupted. Examples include attempting to perform an invalid memory access, external interrupts, and undefined instructions. When an exception occurs, normal program flow is interrupted and execution is resumed at the corresponding exception vector. This contains the first instruction of the interrupt handler to deal with the exception.

Exceptional state	When a potentially exceptional instruction is issued, the FPU coprocessor sets the EX bit, FPEXC[31], and loads a copy of the potentially exceptional instruction in the FPINST register. If the instruction is a short vector operation, the register fields in FPINST are altered to point to the potentially exceptional iteration. When in the exceptional state, the issue of a trigger instruction to the FPU coprocessor causes a bounce. *See also* Bounce, Potentially exceptional instruction, and Trigger instruction.
Exception service routine	*See* Interrupt handler.
Exception vector	*See* Interrupt vector.
Exponent	The component of a floating-point number that normally signifies the integer power to which two is raised in determining the value of the represented number.
External Abort	An indication from an external memory system to a core that the value associated with a memory access is invalid. An external abort is caused by the external memory system as a result of attempting to access invalid memory. *See also* Abort, Data Abort and Prefetch Abort.
Fd	The destination register and the accumulate value in triadic operations. Sd for single-precision operations and Dd for double-precision.
Flush-to-zero mode	In this mode, the FPU coprocessor treats the following values as positive zeros: • arithmetic operation inputs that are in the subnormal range for the input precision • arithmetic operation results, other than computed zero results, that are in the subnormal range for the input precision before rounding. The FPU coprocessor does not interpret these values as subnormal values or convert them to subnormal values. The subnormal range for the input precision is $-2^{Emin} < x < 0$ or $0< x < 2^{Emin}$.
Fm	The second source operand in dyadic or triadic operations. Sm for single-precision operations and Dm for double-precision
Fn	The first source operand in dyadic or triadic operations. Sn for single-precision operations and Dn for double-precision.
Fraction	The floating-point field that lies to the right of the implied binary point.
Halfword	A 16-bit data item.
IEEE 754 standard	*IEEE Standard for Binary Floating-Point Arithmetic, ANSI/IEEE Std. 754-1985*. The standard that defines data types, correct operation, exception types and handling, and error bounds for floating-point systems. Most processors are built in compliance with the standard in either hardware or a combination of hardware and software.
IGN	*See* Ignore.
Ignore (IGN)	Must ignore memory writes.
Illegal instruction	An instruction that is architecturally Undefined.
Implementation-defined	The behavior is not architecturally defined, but is defined and documented by individual implementations.

Implementation-specific	The behavior is not architecturally defined, and does not have to be documented by individual implementations. Used when there are a number of implementation options available and the option chosen does not affect software compatibility.
Infinity	In the IEEE 754 standard format to represent infinity, the exponent is the maximum for the precision and the fraction is all zeros.
Input exception	A FPU exception condition in which one or more of the operands for a given operation are not supported by the hardware. The operation bounces to support code for processing.
Intermediate result	An internal format used to store the result of a calculation before rounding. This format can have a larger exponent field and fraction field than the destination format.
Interrupt handler	A program that control of the processor is passed to when an interrupt occurs.
Interrupt vector	One of a number of fixed addresses in low memory, or in high memory if high vectors are configured, that contains the first instruction of the corresponding interrupt handler.
Load/store architecture	A processor architecture where data-processing operations only operate on register contents, not directly on memory contents.
Load Store Unit (LSU)	The part of a processor that handles load and store transfers.
LSU	*See* Load Store Unit.
Memory bank	One of two or more parallel divisions of interleaved memory, usually one word wide, that enable reads and writes of multiple words at a time, rather than single words. All memory banks are addressed simultaneously and a bank enable or chip select signal determines which of the banks is accessed for each transfer. Accesses to sequential word addresses cause accesses to sequential banks. This enables the delays associated with accessing a bank to occur during the access to its adjacent bank, speeding up memory transfers.
NaN	Not a number. A symbolic entity encoded in a floating-point format that has the maximum exponent field and a nonzero fraction. An SNaN causes an invalid operand exception if used as an operand and a most significant fraction bit of zero. A QNaN propagates through almost every arithmetic operation without signaling exceptions and has a most significant fraction bit of one.
Potentially exceptional instruction	An instruction that is determined, based on the exponents of the operands and the sign bits, to have the potential to produce an overflow, underflow, or invalid condition. After this determination is made, the instruction that has the potential to cause an exception causes the FPU coprocessor to enter the exceptional state and bounce the next trigger instruction issued. *See also* Bounce, Trigger instruction, and Exceptional state.
Processor	A processor is the circuitry in a computer system required to process data using the computer instructions. It is an abbreviation of microprocessor. A clock source, power supplies, and main memory are also required to create a minimum complete working computer system.
Read	Reads are defined as memory operations that have the semantics of a load. That is, the ARM instructions LDM, LDRD, LDC, LDR, LDRT, LDRSH, LDRH, LDRSB, LDRB, LDRBT, LDREX, RFE, STREX, SWP, and SWPB, and the Thumb instructions LDM, LDR, LDRSH, LDRH, LDRSB, LDRB, and POP. Java bytecodes that are accelerated by hardware can cause a number of reads to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.

Reserved	A field in a control register or instruction format is reserved if the field is to be defined by the implementation, or produces Unpredictable results if the contents of the field are not zero. These fields are reserved for use in future extensions of the architecture or are implementation-specific. All reserved bits not used by the implementation must be written as 0 and read as 0.
Rounding mode	The IEEE 754 standard requires all calculations to be performed as if to an infinite precision. For example, a multiply of two single-precision values must accurately calculate the significand to twice the number of bits of the significand. To represent this value in the destination precision, rounding of the significand is often required. The IEEE 754 standard specifies four rounding modes. In round-to-nearest mode, the result is rounded at the halfway point, with the tie case rounding up if it would clear the least significant bit of the significand, making it even. Round-towards-zero mode chops any bits to the right of the significand, always rounding down, and is used by the C, C++, and Java languages in integer conversions. Round-towards-plus-infinity mode and round-towards-minus-infinity mode are used in interval arithmetic.
Saved Program Status Register (SPSR)	The register that holds the CPSR of the task immediately before the exception occurred that caused the switch to the current mode.
SBO	*See* Should Be One.
SBZ	*See* Should Be Zero.
SBZP	*See* Should Be Zero or Preserved.
Scalar operation	A FPU coprocessor operation involving a single source register and a single destination register. *See also* Vector operation.
Short vector operation	A FPU coprocessor operation involving more than one destination register and perhaps more than one source register in the generation of the result for each destination.
Should Be One (SBO)	Should be written as 1 (or all 1s for bit fields) by software. Writing a 0 produces Unpredictable results.
Should Be Zero (SBZ)	Should be written as 0 (or all 0s for bit fields) by software. Writing a 1 produces Unpredictable results.
Should Be Zero or Preserved (SBZP)	Should be written as 0 (or all 0s for bit fields) by software, or preserved by writing the same value back that has been previously read from the same field on the same processor.
Significand	The component of a binary floating-point number that consists of an explicit or implicit leading bit to the left of the implied binary point and a fraction field to the right.
SPSR	*See* Saved Program Status Register
Stride	The stride field, FPSCR[21:20], specifies the increment applied to register addresses in short vector operations. A stride of 00, specifying an increment of +1, causes a short vector operation to increment each vector register by +1 for each iteration, while a stride of 11 specifies an increment of +2.
Subnormal value	A value in the range ($-2^{Emin} < x < 2^{Emin}$), except for 0. In the IEEE 754 standard format for single-precision and double-precision operands, a subnormal value has a zero exponent and a nonzero fraction field. The IEEE 754 standard requires that the generation and manipulation of subnormal operands be performed with the same precision as normal operands.

Support code	Software that must be used to complement the hardware to provide compatibility with the IEEE 754 standard. The support code has a library of routines that performs supported functions, such as divide with unsupported inputs or inputs that might generate an exception in addition to operations beyond the scope of the hardware. The support code has a set of exception handlers to process exceptional conditions in compliance with the IEEE 754 standard.
Tiny	A nonzero result or value that is between the positive and negative minimum normal values for the destination precision.
Trap	An exceptional condition in a FPU coprocessor that has the respective exception enable bit set in the FPSCR register. The user trap handler is executed.
Trigger instruction	The FPU coprocessor instruction that causes a bounce at the time it is issued. A potentially exceptional instruction causes the FPU coprocessor to enter the exceptional state. A subsequent instruction, unless it is an FMXR or FMRX instruction accessing the FPEXC, FPINST, or FPSID register, causes a bounce, beginning exception processing. The trigger instruction is not necessarily exceptional, and no processing of it is performed. It is retried at the return from exception processing of the potentially exceptional instruction. *See also* Bounce, Potentially exceptional instruction, and Exceptional state.
Unaligned	A data item stored at an address that is not divisible by the number of bytes that defines the data size is said to be unaligned. For example, a word stored at an address that is not divisible by four.
Undefined	Indicates an instruction that generates an Undefined instruction trap. See the *ARM Architecture Reference Manual* for more details on ARM exceptions.
UNP	*See* Unpredictable.
Unpredictable	For reads, the data returned when reading from this location is unpredictable. It can have any value. For writes, writing to this location causes unpredictable behavior, or an unpredictable change in device configuration. Unpredictable instructions must not halt or hang the processor, or any part of the system.
Unsupported values	Specific data values that are not processed by the FPU coprocessor hardware but bounced to the support code for completion. These data can include infinities, NaNs, subnormal values, and zeros. An implementation is free to select which of these values is supported in hardware fully or partially, or requires assistance from support code to complete the operation. Any exception resulting from processing unsupported data is trapped to user code if the corresponding exception enable bit for the exception is set.
Vector operation	An FPU coprocessor operation involving more than one destination register, perhaps involving different source registers in the generation of the result for each destination. *See also* Scalar operation.
Word	A 32-bit data item.
Write	Writes are defined as operations that have the semantics of a store. That is, the ARM instructions `SRS`, `STM`, `STRD`, `STC`, `STRT`, `STRH`, `STRB`, `STRBT`, `STREX`, `SWP`, and `SWPB`, and the Thumb instructions `STM`, `STR`, `STRH`, `STRB`, and `PUSH`. Java bytecodes that are accelerated by hardware can cause a number of writes to occur, according to the state of the Java stack and the implementation of the Java hardware acceleration.