Cortex™-A9 Floating-Point Unit

Revision: r4p0

Technical Reference Manual

ARM®

ARM DDI 0408H (ID032812)

Cortex-A9 Floating-Point Unit
Technical Reference Manual

Release Information

The following changes have been made to this book.

Change history

Date	Issue	Confidentiality	Change
04 April 2008	A	Non-Confidential	First release for r0p0
10 July 2008	B	Non-Confidential Restricted Access	Second release for r0p0
12 Dec 2008	C	Non-Confidential Restricted Access	First release for r1p0
28 September 2009	D	Non-Confidential Restricted Access	First release for r2p0
27 November 2009	E	Non-Confidential Unrestricted Access	Second release for r2p0
28 April 2010	F	Non-Confidential Unrestricted Access	First release for r2p2
20 July 2011	G	Non-Confidential	First release for r3p0
23 March 2012	H	Non-Confidential	First release for r4p0

Proprietary Notice

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.

Product Status

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

Web Address

`http://www.arm.com`

Contents
Cortex-A9 Floating-Point Unit Technical Reference Manual

Preface

This preface introduces the *Cortex-A9 Floating-Point Unit (FPU) Technical Reference Manual*. It contains the following sections:

About this book

This book is for the *Cortex-A9 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.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

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

Chapter 2 *Programmers Model*

Read this for a description of the major components of the FPU and how they operate.

Appendix A *Revisions*

Read this for a description of 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 *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 conventions*

Typographical conventions

The typographical conventions are:

italic 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.

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 >	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	Applies when the relevant term is used in body text. For example: IMPLEMENTATION DEFINED, IMPLEMENTATION SPECIFIC, UNKNOWN, and UNPREDICTABLE.

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-A9 Technical Reference Manual* (ARM DDI 0388)
- *Cortex-A9 MPCore Technical Reference Manual* (ARM DDI 0407)
- *Cortex-A9 NEON Media Processing Engine Technical Reference Manual* (ARM DDI 0409)
- *Cortex-A9 MBIST Controller Technical Reference Manual* (ARM DDI 0414)
- *Cortex-A9 Configuration and Sign-Off Guide* (ARM DII 0146)
- *CoreSight™ PTM™-A9 Technical Reference Manual* (ARM DDI 0401)
- *CoreSight PTM-A9 Configuration and Sign-Off Guide* (ARM DII 0161)
- *CoreSight PTM-A9 Integration Manual* (ARM DII 0162)
- *CoreSight Program Flow Trace Architecture Specification* (ARM IHI 0035)
- *ARM® Architecture Reference Manual, ARMv7-A and ARMv7-R edition* (ARM DDI 0406)
- *Application Note 98, VFP Support Code* (ARM DAI 0098)
- *RealView™ Compilation Tools Developer Guide* (ARM DUI 0203)
- *RealView ICE and RealView Trace User Guide* (ARM DUI 0155)
- *Intelligent Energy Controller Technical Overview* (ARM DTO 0005)
- *AMBA® AXI Protocol Specification* (ARM IHI 0022)
- *AMBA Specification* (ARM IHI 0011)
- AMBA Level 2 Cache Controller (L2C-310) Technical Reference Manual (ARM DDI 0246)
- *AMBA Level 2 Cache Controller (L2C-310) Technical Reference Manual* (ARM DDI 0329).

Other publications

This section lists relevant documents published by third parties:

- *ANSI/IEEE Std 754-1985, IEEE Standard for Binary 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 0408H
- the page numbers to which your comments apply
- 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
Introduction

This chapter introduces the FPU. It contains the following sections:

1.1 About the FPU

The FPU is a VFPv3-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
- operation latencies reduced for most operations in single-precision and double-precision
- high data transfer bandwidth through 64-bit split load and store buses
- completion of load transfers can be performed out-of-order
- normalized and denormalized data are all handled in hardware
- trapless operation enabling fast execution
- support for speculative execution
- low power consumption with high level clock gating and small die size.

The FPU fully supports single-precision and double-precision add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also provides conversions between floating-point data formats and ARM integer word format, with special operations to perform the conversion in round-towards-zero mode for high-level language support.

The Cortex-A9 FPU provides an optimized solution in performance, power, and area for embedded applications and high performance for general-purpose applications.

The use of VFP vector mode is deprecated in ARMv7. Vector operations are not supported in hardware. If you use vectors, support code is required. See the *ARM Architecture Reference Manual* for more information.

Note

This manual describes only specific implementation issues. See the *ARM Architecture Reference Manual* for information on the VFPv3 architecture including the instruction set.

1.2 Applications

The 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 Writing optimal FP code

The following guidelines provide significant performance increases for *Floating-Point* (FP) code:

- Moves to and from control registers are serializing. Avoid placing these in loops or time-critical code.
- Avoid register transfers between the Cortex-A9 compute engine register bank and the FPU register bank. Each of the register banks can be loaded or stored directly to or from main memory.
- Avoid too many direct dependencies between consecutive operations. Interleave disparate operations to reduce interlock cycles.
- Avoid the use of single load or store operations and use load and store multiple operations as much as possible to get an efficient transfer bandwidth.
- Perform floating-point compare operations in the FPU and not in the Cortex-A9 processor.

1.4 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 - r1p0 There are no functionality changes although you must use the Cortex-A9 revision r1p0 design with revision r1p0 FPU.

r1p0 - r2p0 There are no functionality changes although you must use the Cortex-A9 revision r2p0 design with this revision r2p0 FPU.

r2p0 - r2p1 There are no functionality changes although you must use the Cortex-A9 revision r2p1 design with this revision r2p1 FPU.

r2p1 - r2p2 There are no functionality changes.

r2p2 - r3p0 There are no functionality changes.

r3p0 - r4p0 There are no functionality changes.

Chapter 2
Programmers Model

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

2.1 About the programmers model

This section introduces the FPU implementation of the VFPv3 floating-point architecture, VFPv3-D16. Unlike VFPv2 implementations, this implementation provides:

- fixed-point to floating-point conversion instructions and floating-point constant loads
- IEEE half-precision and alternative half-precision format support
- trapless exception support.

Table 2-2 on page 2-7 describes the following access type:

RW Read and write.

RO Read only.

2.2 IEEE 754 standard compliance

This section introduces issues related to the IEEE 754 standard compliance:

- hardware and software components
- software-based components and their availability.

2.2.1 Implementation of the IEEE 754 standard

The following operations from the IEEE 754 standard are not supplied by the FPU instruction set:

- remainder
- round floating-point number to integer-valued floating-point number
- binary-to-decimal conversions
- decimal-to-binary conversions
- direct comparison of single-precision and double-precision values.

2.2.2 IEEE 754 standard implementation choices

Some of the implementation choices permitted by the IEEE 754 standard and used in the VFPv3 architecture are described in the *ARM Architecture Reference Manual*.

Supported formats

The VFP supports:

- Single-precision and double-precision for all operations
  - no extended format is supported.
- Half-precision formats
  - IEEE half-precision
  - alternative half-precision.
- Integer formats:
  - unsigned 32-bit integers
  - two’s complement signed 32-bit integers.

2.3 Instruction throughput and latency

Complex instruction dependencies and memory system interactions make it impossible to describe the exact cycle timing of all instructions in all circumstances. The timing described in Table 2-1 on page 2-5 is accurate in most cases. For precise timing, you must use a cycle-accurate model of your processor.

2.3.1 Definitions of throughput and latency

The definitions of throughput and latency are:

Throughput Throughput is the number of cycles after issue that another instruction can begin execution.

Latency Latency is the number of cycles before the data is available for another operation. The forward latency, Fwd, is relevant for *Read After Write* (RAW) hazards. The writeback latency, Wbck, is relevant for *Write-After-Write* (WAW) hazards. See Table 2-1 on page 2-5.

Latency values assume that the instruction has been issued and that neither the FPU pipeline nor the Cortex-A9 pipeline is stalled.

2.3.2 Instruction throughput and latency

Table 2-1 on page 2-5 shows:

- the FPU instruction throughput and latency cycles for all operations except loads, stores and system register accesses
- the old ARM assembler mnemonics and the ARM *Unified Assembler Language* (UAL) mnemonics.

Table 2-1 FPU instruction throughput and latency cycles

Old ARM assembler mnemonic	UAL	Single Precision			Double Precision		
		Throughput	Latency		Throughput	Latency	
			Fwd	Wbck		Fwd	Wbck
FADD FSUB FCVT FSHTOD, FSHTOS FSITOD, FSITOS FTOSHD, FTOSHS FTOSID, FTOSIS FTOSL, FTOUH FTOUI{Z}D, FTOUI{Z}S FTOULD, FTOULS, FUHTOD, FUHTOS FUITOD, FUITOS FULTOD, FULTOS	VADD VSUB VCVT	1	4		1	4	
FMUL FNMUL	VMUL VNMUL	1	5		2	6	
FMAC FNMAC FMSC FNMSC	VMLA VMLS VNMLS VNMLA	1	8		2	9	
FCPY FABS FNEG FCONST	VMOV VABS VNEG VMOV	1	1	2	1	1	2
FMRS FMRR(S/D) FMRD(L/H)	VMOV[a]	1	-	3[b]	1	-	3[b]
FMSR FM(S/D)RR FMD(L/H)R	VMOV[c]	1	1	2	1	1	2
FMSTAT	VMRS	1	-	3[b]	1	-	3[b]
FDIV	VDIV	10	15		20	25	
FSQRT	VSQRT	13	17		28	32	
FCMP FCMPE FCMPZ FCMPEZ	VCMP VCMP{E} VCMP{E} VCMP{E}	1	1	4	1	1	4
-	FCVT(T/B) .F16.F32	1	2	2	-	-	-
-	FCVT(T/B) .F32.F16	1	-	4	-	-	-

a. FPU to ARM.
b. The writeback number for these instructions is given from an ARM core writeback point of view. It reflects the penalty of moving data from the FPU into the ARM core register file before the following ARM instruction can use the moved data.
c. ARM to FPU.

2.4 Register summary

Table 2-2 shows the FPU system registers. All FPU system registers are 32-bit wide. Reserved register addresses are RAZ/WI.

Table 2-2 FPU system registers

Name	Type	Reset	Description
FPSID	RO	0x41033094	See *Floating-Point System ID Register* on page 2-9
FPSCR	RW	0x00000000	See *Floating-Point Status and Control Register* on page 2-9
MVFR1	RO	0x01000011[a]	See the *ARM Architecture Reference Manual*
MVFR0	RO	0x10110221[a]	Seethe *ARM Architecture Reference Manual*
FPEXC	RW	0x00000000	See *Floating-Point Exception Register* on page 2-11

a. The reset values in this document are valid for an FPU-only configuration of the Cortex-A9.
In Cortex-A9 configurations with the Media Processing Engine, SIMD Extensions, the MVFR0 and MVFR1 values are different. See the *Cortex-A9 NEON Media Processing Engine Technical Reference Manual* for more information.

2.4.1 Processor modes for accessing the FPU system registers

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

Table 2-3 Accessing 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

2.4.2 Accessing the FPU registers

Access to the FPU registers is controlled by two system control coprocessor registers of the Cortex-A9 processor, accessed through CP15:

- *Non-secure Access Control Register* (NSACR)
- *Coprocessor Access Control Register* (CPACR).

See the *Cortex-A9 Technical Reference Manual* for information on these registers.

To use the FPU in Secure state only

To use the FPU in Secure state only, you must define the CPACR and Floating-Point Exception Register (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, you must 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/neon
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
```

2.5 Register descriptions

This section describes the FPU system registers. Table 2-2 on page 2-7 provides cross references to individual registers.

2.5.1 Floating-Point System ID Register

The FPSID Register characteristics are:

Purpose Provides information about the VFP implementation.

Usage constraints Only accessible in privileged modes.

Configurations Available in all FPU configurations.

Attributes See the register summary in Table 2-2 on page 2-7.

Figure 2-1 shows the FPSID Register bit assignments.

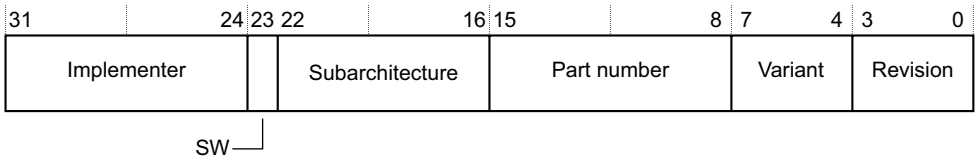

Figure 2-1 FPSID Register bit assignments

Table 2-4 shows the FPSID Register bit assignments.

Table 2-4 FPSID Register bit assignments

Bits	Name	Function
[31:24]	Implementer	Denotes ARM
[23]	SW	Hardware implementation with no software emulation
[22:16]	Subarchitecture	The null VFP sub-architecture
[15:8]	Part number	VFPv3
[7:4]	Variant	Cortex-A9
[3:0]	Revision	Revision 4

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

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

2.5.2 Floating-Point Status and Control Register

The FPSCR characteristics are:

Purpose Provides User-level control of the FPU.

Usage constraints There are no usage constraints.

Configurations Available in all FPU configurations.

Attributes See the register summary in Table 2-2 on page 2-7.

Figure 2-2 on page 2-10 shows the FPSCR bit assignments.

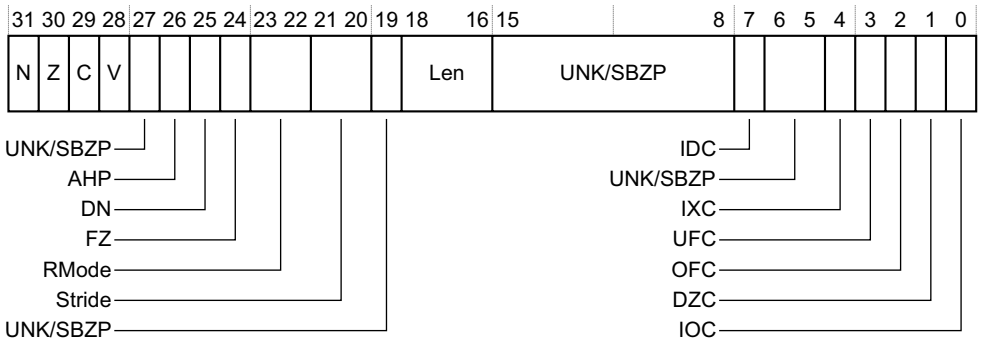

Figure 2-2 FPSCR bit assignments

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]	-	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. Advanced SIMD arithmetic always uses the Default NaN setting, regardless of the value of the DN bit.
[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. Advanced SIMD arithmetic always uses the Flush-to-zero setting, regardless of the value of the FZ bit.
[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. Advanced SIMD arithmetic always uses the Round to nearest setting, regardless of the value of the RMode bits.
[21:20]	Stride	Stride control used for backwards compatibility with short vector values. See the *ARM Architecture Reference Manual*.
[19]	-	UNK/SBZP.
[18:16]	Len	Vector length, used for backwards compatibility with short vector values. See the *ARM Architecture Reference Manual*.

Table 2-5 FPSCR bit assignments (continued)

Bits	Name	Function
[15:8]	-	UNK/SBZP.
[7]	IDC	Input Denormal cumulative exception flag.[a]
[6:5]	-	UNK/SBZP.
[4]	IXC	Inexact cumulative exception flag.[a]
[3]	UFC	Underflow cumulative exception flag.[a]
[2]	OFC	Overflow cumulative exception flag.[a]
[1]	DZC	Division by Zero cumulative exception flag.[a]
[0]	IOC	Invalid Operation cumulative exception flag.[a]

a. The exception flags, bit [7] and bits [4:0] of the FPSCR are exported on the **DEFLAGS** output so they can be monitored externally to the processor, if required.

You can access the FPSCR Register 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
```

2.5.3 Floating-Point Exception Register

The FPEXC Register characteristics are:

Purpose Provides global enable control of the Advanced SIMD and VFP extensions.

Usage constraints Accessible in all FPU configurations, with restrictions. See *Processor modes for accessing the FPU system registers* on page 2-7.

Configurations Available in all FPU configurations.

Attributes See the register summary in Table 2-2 on page 2-7.

Figure 2-3 shows the FPEXC Register bit assignments.

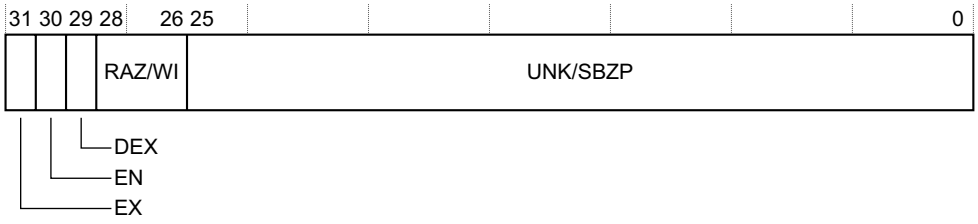

Figure 2-3 FPEXC Register bit assignments

Table 2-6 shows the FPEXC Register bit assignments.

Table 2-6 FPEXC Register bit assignments

Bits	Name	Function
[31]	EX	Exception bit: This bit reads-as-zero and ignores writes. The Cortex-A9 NEON MPE never requires asynchronous exception handling.
[30]	EN	Enable bit: **0** VFP extension is disabled. **1** VFP extension is enabled and operates normally. The EN bit is cleared to 0 at reset.
[29]	DEX	Defined synchronous instruction exceptional flag: **0** No exception has occurred. **1** Attempt to perform a VFP vector operation has been trapped.[a] The DEX bit is cleared to 0 at reset.
[28:26]	-	RAZ/WI.
[25:0]	-	UNK/SBZP.

a. The Cortex-A9 FPU hardware does not support the deprecated VFP short vector feature. Attempts to execute VFP data-processing instructions when the FPSCR.LEN field is non-zero result in the FPSCR.DEX bit being set and a synchronous Undefined instruction exception being taken. You can use software to emulate the short vector feature, if required.

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
First release	-

Table A-2 Differences between Issue A and Issue B

Change	Location
UAL instructions to access registers are added to register descriptions	*Register descriptions* on page 2-9

Table A-3 Differences between issue B and issue C

Change	Location
The mnemonic for the `FMXR` instruction is changed to `VMSR`	*To use the FPU in Secure state only* on page 2-8 and *To use the FPU in Secure state and Non-secure state* on page 2-8
Updated FPSCR bit assignments table	Table 2-5 on page 2-10

Table A-4 Differences between issue C and issue D

Change	Location
Front matter	*Preface* on page iv
Revision number updates	Table 2-2 on page 2-7 and Table 2-4 on page 2-9
SBZ replaced with UNK/SBZP	Fig 2-3 and Table 2-6, Fig 2-5 and Table 3-8

Table A-5 Differences between issue D and Issue E

Change	Location
No technical change	-

Table A-6 Differences between issue E and Issue F

Change	Location
Harmonized FPEXC bit descriptions with the MPE TRM descriptions	Table 2-6 on page 2-12

Table A-7 Differences between issue F and issue G

Change	Location	Affects
Updated latency information for `VMOV` and `VMRS` instructions	Table 2-1 on page 2-5	All versions
Updated information about reset values for the MVFR1 and MVFR0 registers	*Register summary* on page 2-7	All versions
Updated revision number	Table 2-2 on page 2-7	r3p0
	Table 2-4 on page 2-9	r3p0

Table A-8 Differences between issue G and Issue H

Change	Location	Affects
Updated revision number	Table 2-2 on page 2-7	r4p0
	Table 2-4 on page 2-9	