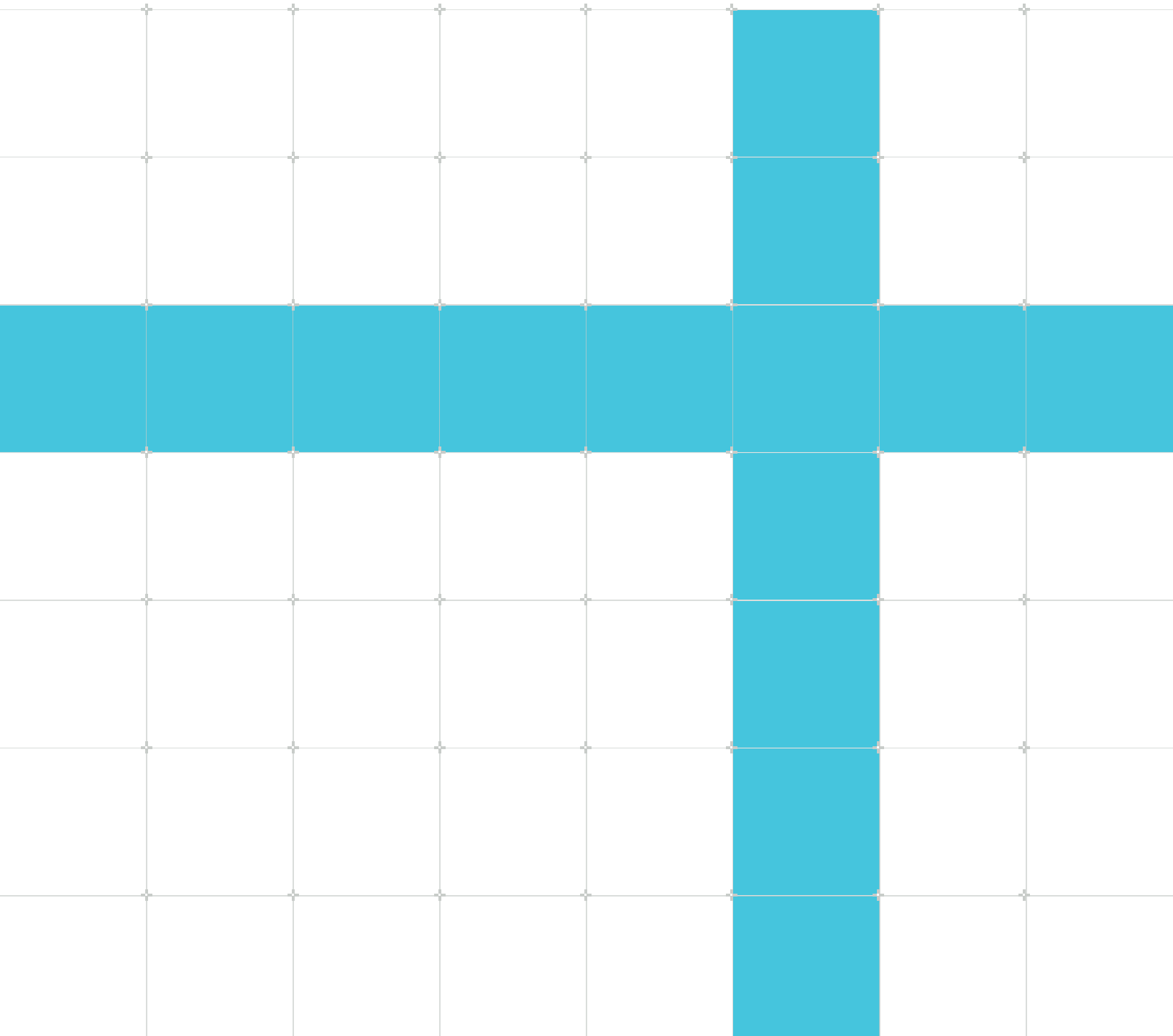
Arm® Cortex®-A78AE Core Cryptographic Extension

Revision: r0p2

Technical Reference Manual

Non-Confidential

Issue 07
101799_0002_07_en



Arm® Cortex®-A78AE Core Cryptographic Extension

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-01	28 June 2019	Confidential	First development release for r0p0
0000-02	31 October 2019	Confidential	Second development release for r0p0
0000-03	31 January 2020	Confidential	First release for r0p0
0001-04	30 April 2020	Confidential	First early access release for r0p1
0001-05	29 September 2020	Non-Confidential	Second early access release for r0p1
0001-06	13 November 2020	Non-Confidential	Third early access release for r0p1
0002-07	31 March 2022	Non-Confidential	First release for r0p2

Proprietary Notice

This document may include technical inaccuracies or typographical errors.

TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, INCLUDING WITHOUT LIMITATION ANY DIRECT, INDIRECT, SPECIAL, INCIDENTAL, PUNITIVE, OR CONSEQUENTIAL DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF ANY USE OF THIS DOCUMENT, EVEN IF ARM HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

This document consists solely of commercial items. You shall be responsible for ensuring that any use, duplication or disclosure of this document complies fully with any relevant export laws and regulations to assure that this document or any portion thereof is not exported, directly or indirectly, in violation of such export laws. Use of the word “partner” in reference to Arm’s customers is not intended to create or refer to any partnership relationship with any other company. Arm may make changes to this document at any time and without notice.

This document may be translated into other languages for convenience, and you agree that if there is any conflict between the English version of this document and any translation, the terms of the English version of the Agreement shall prevail.

Arm Limited. Company 02557590 registered in England.

110 Fulbourn Road, Cambridge, England CB1 9NJ.

(LES-PRE-20349)

Confidentiality Status

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

Unrestricted Access is an Arm internal classification.

Product Status

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

Feedback

Arm welcomes feedback on this product and its documentation. To provide feedback on the product, create a ticket on https://support.developer.arm.com.

To provide feedback on the document, fill the following survey: https://developer.arm.com/documentation-feedback-survey.

Inclusive language commitment

Arm values inclusive communities. Arm recognizes that we and our industry have used language that can be offensive. Arm strives to lead the industry and create change.

This document includes language that can be offensive. We will replace this language in a future issue of this document.

To report offensive language in this document, email terms@arm.com.

Contents

1 Introduction

1.1 Product revision status

The r*x*p*y* identifier indicates the revision status of the product described in this manual, for example, r1p2, where:

r*x* Identifies the major revision of the product, for example, r1.
p*y* Identifies the minor revision or modification status of the product, for example, p2.

1.2 Intended audience

This manual is for system designers, system integrators, and programmers who are designing or programming a *System-on-Chip* (SoC) that uses the Cortex®-A78AE core with the optional Cryptographic Extension.

1.3 Conventions

The following subsections describe conventions used in Arm documents.

Glossary

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

See the Arm Glossary for more information: developer.arm.com/glossary.

Typographic conventions

Convention	Use
italic	Citations.
bold	Interface elements, such as menu names. Signal names. Terms in descriptive lists, where appropriate.
`monospace`	Text that you can enter at the keyboard, such as commands, file and program names, and source code.
`monospace bold`	Language keywords when used outside example code.
`monospace underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

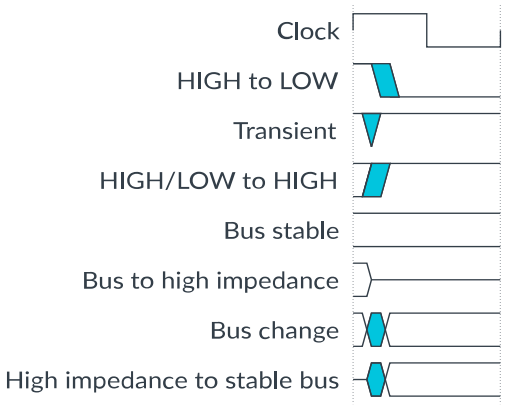
Convention	Use
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.
Caution	Recommendations. Not following these recommendations might lead to system failure or damage.
Warning	Requirements for the system. Not following these requirements might result in system failure or damage.
Danger	Requirements for the system. Not following these requirements will result in system failure or damage.
Note	An important piece of information that needs your attention.
Tip	A useful tip that might make it easier, better or faster to perform a task.
Remember	A reminder of something important that relates to the information you are reading.

Timing diagrams

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

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

Figure 1-1: Key to timing diagram conventions



Signals

The signal conventions are:

Signal level

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

- HIGH for active-HIGH signals.
- LOW for active-LOW signals.

Lowercase n

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

1.4 Additional reading

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

Table 1-2: Arm publications

Document name	Document ID	Licensee only
Arm® Cortex®-A78AE Core Technical Reference Manual	101779	No
Arm® Cortex®-A78AE Core Configuration and Integration Manual	101780	Yes
Arm® Architecture Reference Manual Armv8, for A-profile architecture	DDI 0487	No

Table 1-3: Other publications

Document ID	Organization	Document name
FIPS 197	-	*Advanced Encryption Standard.*
FIPS 180-4	-	*Secure Hash Standard (SHS)*

Arm tests its PDFs only in Adobe Acrobat and Acrobat Reader. Arm cannot guarantee the quality of its documents when used with any other PDF reader.

Adobe PDF reader products can be downloaded at http://www.adobe.com

2 Functional description

This chapter describes the Cortex®-A78AE core Cryptographic Extension.

2.1 About the Cryptographic Extension

The Cortex®-A78AE core Cryptographic Extension supports the Arm®v8-A Cryptographic Extension. Some parts of the Arm®v8-A Cryptographic Extension are optional.

For more information on the optional parts of the Arm®v8-A Cryptographic Extension, see the *AArch64 Instruction Set Attribute Register 0, EL1* register (ID_AA64ISAR0_EL1) in the *Arm® Cortex®-A78AE Core Technical Reference Manual*.

The Cryptographic Extension adds new A64, A32, and T32 instructions to Advanced SIMD that accelerate *Advanced Encryption Standard* (AES) encryption and decryption. It also adds instructions to implement the *Secure Hash Algorithm* (SHA) functions SHA-1, SHA-224, and SHA-256.

The optional Cryptographic Extension is not included in the base product. Arm supplies the Cryptographic Extension only under an additional license to the Cortex®-A78AE core.

2.2 Revisions

This section describes the differences in functionality between product revisions.

r0p0 First release

r0p1 No functional changes

r0p2 No functional changes

3 Register descriptions

This chapter describes the Cryptographic Extension registers.

3.1 Identifying the Cryptographic instructions implemented

Software can identify the Cryptographic instructions that are implemented by reading two registers.

About this task

The two registers are:

- ID_AA64ISAR0_EL1 in the AArch64 Execution state
- ID_ISAR5_EL1 in the AArch64 Execution state

3.2 Disabling the Cryptographic Extension

To disable the Cryptographic Extension, assert the **CRYPTODISABLE** input signal, which applies to all the Cortex®-A78AE cores present in a cluster. This signal is sampled only during reset of the cores.

About this task

When **CRYPTODISABLE** is asserted:

- Executing a Cryptographic instruction results in an **UNDEFINED** exception.
- The ID registers described in Cryptographic Extension register summary on page 10 indicate that the Cryptographic Extension is not implemented.

3.3 Register summary

The core has two instruction identification registers. Each register has a specific purpose, usage constraints, configurations, and attributes.

The following table lists the instruction identification registers for the Cortex®-A78AE core Cryptographic Extension.

Table 3-1: Cryptographic Extension register summary

Name	Execution state	Description
ID_AA64ISAR0_EL1	AArch64	See 3.4 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1 on page 11.
ID_ISAR5_EL1	AArch64	See 3.5 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1 on page 13.

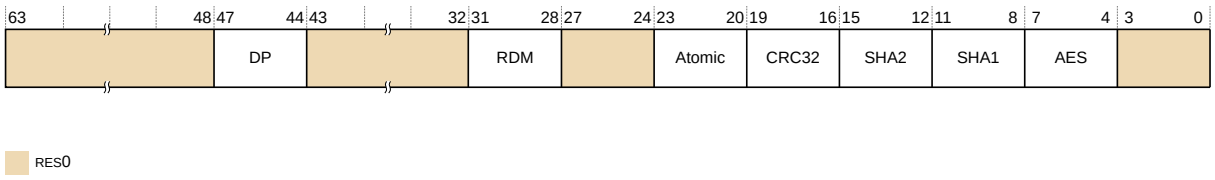
3.4 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0, EL1

The ID_AA64ISAR0_EL1 provides information about the instructions that are implemented in AArch64 state, including the instructions provided by the Cryptographic Extension.

Bit field descriptions

ID_AA64ISAR0_EL1 is a 64-bit register.

Figure 3-1: ID_AA64ISAR0_EL1 bit assignments



RES0, [63:48]

RES0	Reserved

DP, [47:44]

Indicates whether Dot Product support instructions are implemented.

0x1	UDOT, SDOT instructions are implemented.

RES0, [43:32]

RES0	Reserved

RDM, [31:28]

Indicates whether *Rounding Double Multiply* (RDM) instructions are implemented. The value is:

0x1	SQRDMLAH and SQRDMLSH instructions are implemented.

RES0, [27:24]

RES0	Reserved

Atomic, [23:20]

Indicates whether atomic instructions are implemented. The value is:

0x2	LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions are implemented.

CRC32, [19:16]

Indicates whether CRC32 instructions are implemented. The value is:

`0x1` CRC32 instructions are implemented.

SHA2, [15:12]

Indicates whether SHA2 instructions are implemented. The possible values are:

`0x0` No SHA2 instructions are implemented. This is the value if the core implementation does not include the Cryptographic Extension.

`0x1` `SHA256H`, `SHA256H2`, `SHA256U0`, and `SHA256U1` are implemented. This is the value if the core implementation includes the Cryptographic Extension.

SHA1, [11:8]

Indicates whether SHA1 instructions are implemented. The possible values are:

`0x0` No SHA1 instructions are implemented. This is the value if the core implementation does not include the Cryptographic Extension.

`0x1` `SHA1C`, `SHA1P`, `SHA1M`, `SHA1SU0`, and `SHA1SU1` are implemented. This is the value if the core implementation includes the Cryptographic Extension.

AES, [7:4]

Indicates whether AES instructions are implemented. The possible values are:

`0x0` No AES instructions implemented. This is the value if the core implementation does not include the Cryptographic Extension.

`0x2` `AESE`, `AESD`, `AESMC`, and `AESIMC` are implemented, plus `PMULL` and `PMULL2` instructions operating on 64-bit data. This is the value if the core implementation includes the Cryptographic Extension.

RES0, [3:0]

RES0 Reserved

Configurations

ID_AA64ISAR0_EL1 is architecturally mapped to external register ID_AA64ISAR0.

Usage constraints

Accessing the ID_AA64ISAR0_EL1

To access the ID_AA64ISAR0_EL1:

```
MRS <Xt>, ID_AA64ISAR0_EL1 ; Read ID_AA64ISAR0_EL1 into Xt
```

Register access is encoded as follows:

Table 3-2: ID_AA64ISAR0_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0110	000

Accessibility

This register is accessible as follows:

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

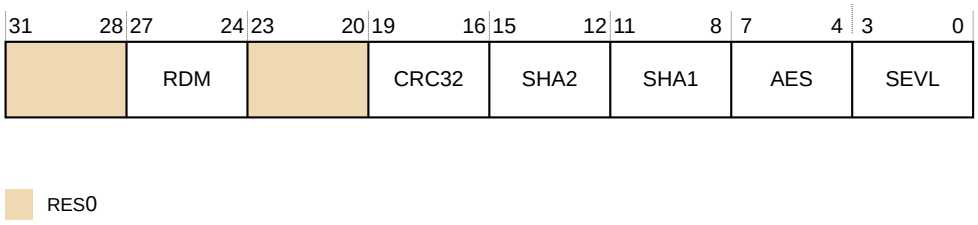
3.5 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5, EL1

The AArch64 register ID_ISAR5_EL1 provides information about the instructions that are implemented in AArch32 state, including the instructions provided by the optional Cryptographic Extension.

Bit-field descriptions

ID_ISAR5_EL1 is a 32-bit register.

Figure 3-2: ID_ISAR5_EL1 bit assignments



RES0, [31:28]

RES0 Reserved

RDM, [27:24]

Indicates whether RDM instructions are implemented. The value is:

`0x1` `SQRDMLAH` and `SQRDMLSH` instructions are implemented.

RES0, [23:20]

RES0 Reserved

CRC32, [19:16]

Indicates whether CRC32 instructions are implemented in AArch32 state. The value is:

`0x1` CRC32 instructions are implemented.

SHA2, [15:12]

Indicates whether SHA2 instructions are implemented in AArch32 state. The possible values are:

`0x0` Cryptographic Extension is not implemented or is disabled.
`0x1` `SHA256H`, `SHA256H2`, `SHA256SU0`, and `SHA256SU1` instructions are implemented.

SHA1, [11:8]

Indicates whether SHA1 instructions are implemented in AArch32 state. The possible values are:

`0x0` Cryptographic Extension is not implemented or is disabled.
`0x1` `SHA1C`, `SHA1P`, `SHA1M`, `SHA1H`, `SHA1SU0`, and `SHA1SU1` instructions are implemented.

AES, [7:4]

Indicates whether AES instructions are implemented in AArch32 state. The possible values are:

`0x0` Cryptographic Extension is not implemented or is disabled.
`0x2` `AESE`, `AESD`, `AESMC`, and `AESIMC` are implemented, plus `PMULL` and `PMULL2` instructions operating on 64-bit data.

SEVL, [3:0]

Indicates whether the `SEVL` instruction is implemented. The value is:

`0x1` `SEVL` implemented to send event local.

Configurations

This register has no configuration options.

Usage constraints

Accessing the ID_ISAR5_EL1

To access the ID_ISAR5_EL1:

```
MRS <Xt>, ID_ISAR5_EL1 ; Read ID_ISAR5_EL1 into Xt
```

Register access is encoded as follows:

Table 3-4: ID_ISAR5_EL1 access encoding

op0	op1	CRn	CRm	op2
11	000	0000	0010	101

Accessibility

This register is accessible as follows:

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

Appendix A Document revisions

Changes between released issues of this book are summarized in tables.

A.1 Revisions

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

Table A-1: Issue 0000-01

Change	Location
First Confidential development release for r0p0	-

Table A-2: Differences between Issue 0000-01 and Issue 0000-02

Change	Location
Second Confidential development release for r0p0	-
No technical changes	-

Table A-3: Differences between Issue 0000-02 and Issue 0000-03

Change	Location
First Confidential release for r0p0	-
No technical changes	-

Table A-4: Differences between Issue 0000-03 and Issue 0001-04

Change	Location
First Confidential early access release for r0p1	-
No technical changes	-

Table A-5: Differences between 0001-04 and 0001-05

Change	Location
Second Non-Confidential early access release for r0p1	-
Updated product name to Cortex®-A78AE	Throughout document.

Table A-6: Differences between 0001-05 and 0001-06

Change	Location
Third Non-Confidential early access release for r0p1	-
No technical changes to this document since last revision	-

Table A-7: Differences between 0001-06 and 0002-07

Change	Location
First Non-Confidential release for r0p2	-
No technical changes to this document since last revision	-