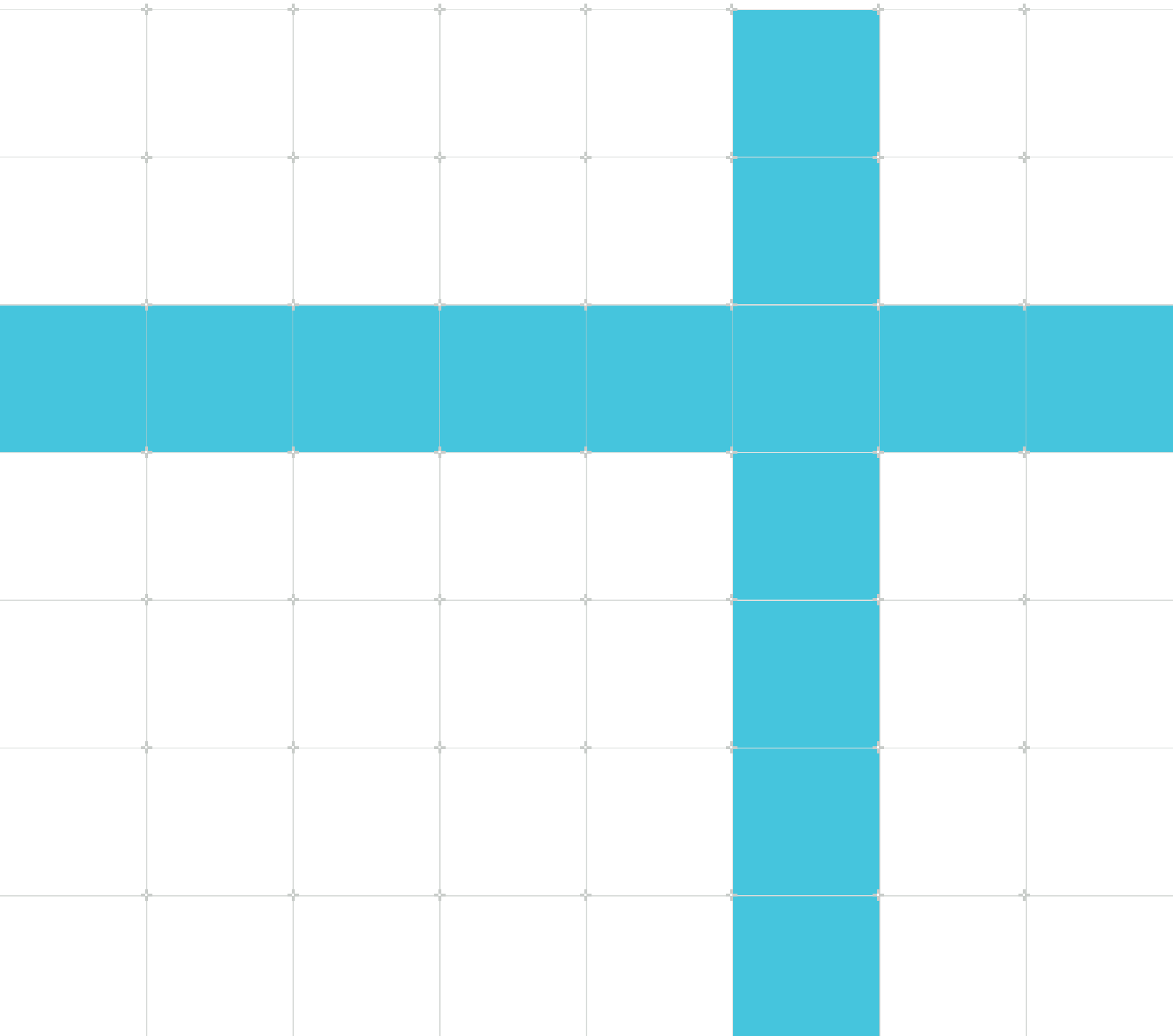
Arm® Cortex®-A710 Core Cryptographic Extension

Revision: r2p1

Technical Reference Manual

Non-Confidential

Issue 06
101802_0201_06_en



Arm® Cortex®-A710 Core Cryptographic Extension

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-01	24 January 2020	Confidential	First beta release for r0p0
0000-02	30 March 2020	Confidential	First limited access release for r0p0
0100-03	12 June 2020	Confidential	First limited access release for r1p0
0200-04	21 August 2020	Confidential	First early access release for r2p0
0200-05	25 May 2020	Non-Confidential	Second early access release for r2p0
0201-06	10 December 2021	Non-Confidential	First release for r2p1

Proprietary Notice

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.

If any of the provisions contained in these terms conflict with any of the provisions of any click through or signed written agreement covering this document with Arm, then the click through or signed written agreement prevails over and supersedes the conflicting provisions of these terms. 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.

Inclusive language commitment

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

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

If you find offensive terms in this document, please contact 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®-A710 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	Introduces 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 bold`	Denotes language keywords when used outside example code.
`monospace underline`	Denotes a permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.

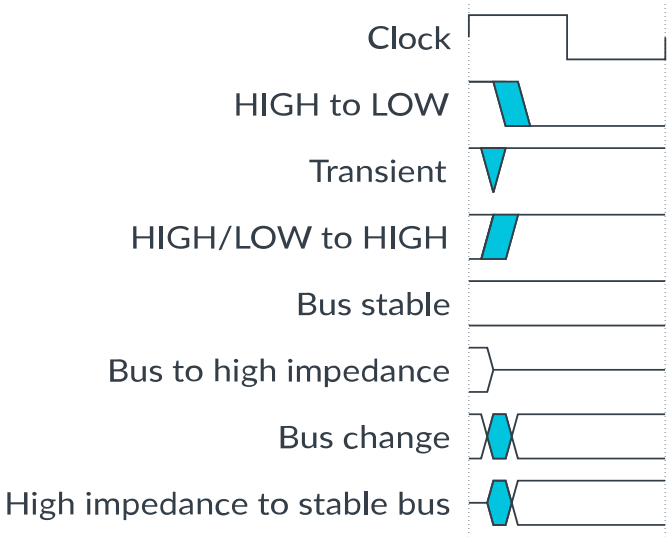
Convention	Use
<and>	Encloses replaceable terms for assembler syntax where they appear in code or code fragments. For example: `MRC p15, 0, <Rd>, <CRn>, <CRm>, <Opcode_2>`
SMALL CAPITALS	Used in body text for a few terms that have specific technical meanings, that are defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.
	This represents a recommendation which, if not followed, might lead to system failure or damage.
	This represents a requirement for the system that, if not followed, might result in system failure or damage.
	This represents a requirement for the system that, if not followed, will result in system failure or damage.
	This represents an important piece of information that needs your attention.
	This represents a useful tip that might make it easier, better or faster to perform a task.
	This is 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®-A710 Core Technical Reference Manual	101800	No
Arm® Cortex®-A710 Core Configuration and Integration Manual	101801	Yes
Arm® Architecture Reference Manual Armv8, for A-profile architecture	DDI 0487	No

Document Name	Document ID	Licensee only
Arm® Architecture Reference Manual Supplement Armv9, for Armv9-A architecture profile	DDI 0608	No

Table 1-3: Other publications

Document Name	Document ID
Advanced Encryption Standard (FIPS 197, November 2001)	-
Secure Hash Standard (SHS) (FIPS 180-4, August 2015)	-
Secure Hash Standard (SHS) (FIPS 202, August 2015)	-

1.5 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

Information about how to give feedback on the content.

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

- The title Arm® Cortex®-A710 Core Cryptographic Extension Technical Reference Manual.
- The number 101802_0201_06_en.
- If applicable, the page number(s) to which your comments refer.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.


Note

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

2 Cryptographic extension support in the Cortex®-A710 core

The Cortex®-A710 core supports the optional Arm®v8.0-A and Arm®v8.2-A Cryptographic Extension.

The Arm®v8.0-A Cryptographic Extension adds A64 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 Arm®v8.2-A extensions, Armv8.2-A-SHA and Armv8.2-SM, add A64 instructions to accelerate SHA2-512, SHA3, SM3, and SM4.

The SVE2-AES, SVE2-SHA3, and SVE2-SM extensions add A64 instructions to accelerate SHA3, SM3, SM4, and AES encryption and decryption.

2.1 Product Revisions

The following table indicates the main differences in functionality between product revisions.

Table 2-1: Product revisions

Revision	Notes
r0p0	First release. There are no technical changes.
r1p0	Second release. There are no technical changes.
r2p0	Third release. There are no technical changes.
r2p1	Fourth release. There are no technical changes.

Changes in functionality that have an impact on the documentation also appear in A.1 Revisions on page 17.

2.2 Disabling the Cryptographic Extension

Disabling of the Cryptographic Extension applies to all Cortex®-A710 cores in a cluster.

To disable the Cryptographic Extension, assert **CRYPTODISABLE**.

When **CRYPTODISABLE** is asserted:

- Executing a cryptographic instruction results in an **UNDEFINED** exception.
- ID_AA64ISAR0_EL1 and ID_ISAR5_EL1 indicates that the Cryptographic Extension is not implemented.

Related information

2.4 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0 on page 11
2.5 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5 on page 14

2.3 Cryptographic Extensions register summary

Software can identify the cryptographic instructions that are implemented in the Cortex®-A710 core by reading identification registers.

The following table shows the instruction identification registers for the Cortex®-A710 core Cryptographic Extension.

Table 2-2: Cryptographic Extension register summary

Name	Execution state	Description
ID_AA64ISAR0_EL1	AArch64	See 2.4 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0 on page 11
ID_ISAR5_EL1	AArch32	See 2.5 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5 on page 14

2.4 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0

Provides information about the instructions implemented in AArch64 state.

For general information about the interpretation of the ID registers, see 'Principles of the ID scheme for fields in ID registers'.

Configurations

- This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure 2-1: AArch64_ID_AA64ISAR0_EL1 bit assignments

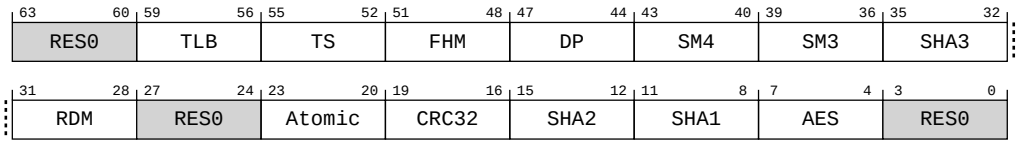


Table 2-3: ID_AA64ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	RES0	Reserved	0b0
[59:56]	TLB	Indicates support for Outer shareable and TLB range maintenance instructions. Defined values are: **0b0010** Outer shareable and TLB range maintenance instructions are implemented.	
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: **0b0010** CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. Defined values are: **0b0001** FMLAL and FMLSL instructions are implemented.	
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. Defined values are: **0b0001** UDOT and SDOT instructions implemented.	
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SM3 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SM3 instructions SM4E and SM4EKEY are implemented.	
[39:36]	SM3	Indicates support for SM3 instructions in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SM4 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SM4 instructions SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 are implemented.	
[35:32]	SHA3	Indicates support for SHA3 instructions in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SHA3 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SHA3 instructions EOR3, RAX1, XAR, and BCAX are implemented.	

Bits	Name	Description	Reset
[31:28]	RDM	Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are: **0b0001** SQRDMLAH and SQRDMLSH instructions implemented.	
[27:24]	RES0	Reserved	0b0
[23:20]	Atomic	Indicates support for Atomic instructions in AArch64 state. Defined values are: **0b0010** LDADD, LDCLR, LDEOR, LDSET, LDSMAX, LDSMIN, LDUMAX, LDUMIN, CAS, CASP, and SWP instructions implemented.	
[19:16]	CRC32	CRC32 instructions implemented in AArch64 state. Defined values are: **0b0001** CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented.	
[15:12]	SHA2	SHA2 instructions implemented in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SHA2 instructions are not implemented. **0b0010** When Cryptographic extensions are implemented and enabled then SHA256H, SHA256H2, SHA256SU0, SHA256SU1, SHA512H, SHA512H2, SHA512SU0, and SHA512SU1 instructions are implemented.	
[11:8]	SHA1	SHA1 instructions implemented in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then SHA1 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions are implemented.	
[7:4]	AES	AES instructions implemented in AArch64 state. Defined values are: **0b0000** When Cryptographic extensions are not implemented or disabled then AES instructions are not implemented. **0b0010** When Cryptographic extensions are implemented and enabled then AESE, AESD, AESMC, and AESIMC instructions are implemented and also PMULL/PMULL2 instructions operating on 64-bit data quantities.	
[3:0]	RES0	Reserved	0b0

Access

MRS <Xt>, ID_AA64ISAR0_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_AA64ISAR0_EL1	0b11	0b000	0b0000	0b0110	0b000

Accessibility

MRS <Xt>, ID_AA64ISAR0_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ISAR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ISAR0_EL1;
```

2.5 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5

Provides information about the instruction sets implemented by the PE in AArch32 state.

Must be interpreted with AArch64-ID_ISAR0_EL1, AArch64-ID_ISAR1_EL1, AArch64-ID_ISAR2_EL1, AArch64-ID_ISAR3_EL1, and AArch64-ID_ISAR4_EL1.

For general information about the interpretation of the ID registers see 'Principles of the ID scheme for fields in ID registers'.

Configurations

- This register is available in all configurations.

Attributes

Width

64

Functional group

Identification

Reset value

See individual bit resets.

Bit descriptions

Figure 2-2: AArch64_ID_ISAR5_EL1 bit assignments

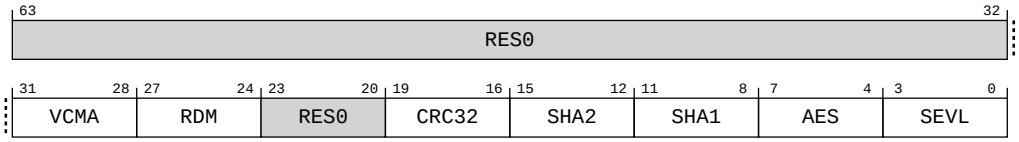


Table 2-5: ID_ISAR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	RES0	Reserved	0b0
[31:28]	VCMA	Indicates AArch32 support for complex number addition and multiplication where numbers are stored in vectors. Defined values are: **0b0001** The VCMLA and VCADD instructions are implemented in AArch32.	
[27:24]	RDM	Indicates whether the VQRDMLAH and VQRDMLSH instructions are implemented in AArch32 state. Defined values are: **0b0001** VQRDMLAH and VQRDMLSH instructions implemented.	
[23:20]	RES0	Reserved	0b0
[19:16]	CRC32	Indicates whether the CRC32 instructions are implemented in AArch32 state. **0b0001** CRC32B, CRC32H, CRC32W, CRC32CB, CRC32CH, and CRC32CW instructions implemented.	
[15:12]	SHA2	Indicates whether the SHA2 instructions are implemented in AArch32 state. **0b0000** When Cryptographic extensions are not implemented or disabled then SHA2 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SHA256H, SHA256H2, SHA256SU0, and SHA256SU1 instructions are implemented.	
[11:8]	SHA1	Indicates whether the SHA1 instructions are implemented in AArch32 state. **0b0000** When Cryptographic extensions are not implemented or disabled then SHA1 instructions are not implemented. **0b0001** When Cryptographic extensions are implemented and enabled then SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions are implemented.	
[7:4]	AES	Indicates whether the AES instructions are implemented in AArch32 state. **0b0000** When Cryptographic extensions are not implemented or disabled then AES instructions are not implemented. **0b0010** When Cryptographic extensions are implemented and enabled then AESE, AESD, AESMC, AESIMC and VMULL.64 instructions are implemented.	

Bits	Name	Description	Reset
[3:0]	SEVL	Indicates whether the SEVL instruction is implemented in AArch32 state. **0b0001** SEVL is implemented as Send Event Local.	

Access

MRS <Xt>, ID_ISAR5_EL1

<systemreg>	op0	op1	CRn	CRm	op2
ID_ISAR5_EL1	0b11	0b000	0b0000	0b0010	0b101

Accessibility

MRS <Xt>, ID_ISAR5_EL1

```
if PSTATE.EL == EL0 then
    if EL2Enabled() && HCR_EL2.TGE == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        AArch64.SystemAccessTrap(EL1, 0x18);
elsif PSTATE.EL == EL1 then
    if EL2Enabled() && HCR_EL2.TID3 == '1' then
        AArch64.SystemAccessTrap(EL2, 0x18);
    else
        return ID_ISAR5_EL1;
elsif PSTATE.EL == EL2 then
    return ID_ISAR5_EL1;
elsif PSTATE.EL == EL3 then
    return ID_ISAR5_EL1;
```

Appendix A Document revisions

This appendix records the changes between released issues of this document.

A.1 Revisions

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

The first table is for the first release. Then, each table compares the new issue of the book with the last released issue of the book. Release numbers match the revision history in Release Information on page 2.

Table A-1: Issue 0000-01

Change	Location
First beta release for r0p0	-

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

Change	Location
First limited access release for r0p0	-

Table A-3: Differences between Issue 0000-02 and Issue 0100-03

Change	Location
First limited access release for r1p0	-

Table A-4: Differences between Issue 0100-03 and Issue 0200-04

Change	Location
First early access release for r2p0	-

Table A-5: Differences between Issue 0200-04 and Issue 0200-05

Change	Location
Second early access release for r2p0	-

Table A-6: Differences between Issue 0200-05 and Issue 0201-06

Change	Location
First release for r2p1	-