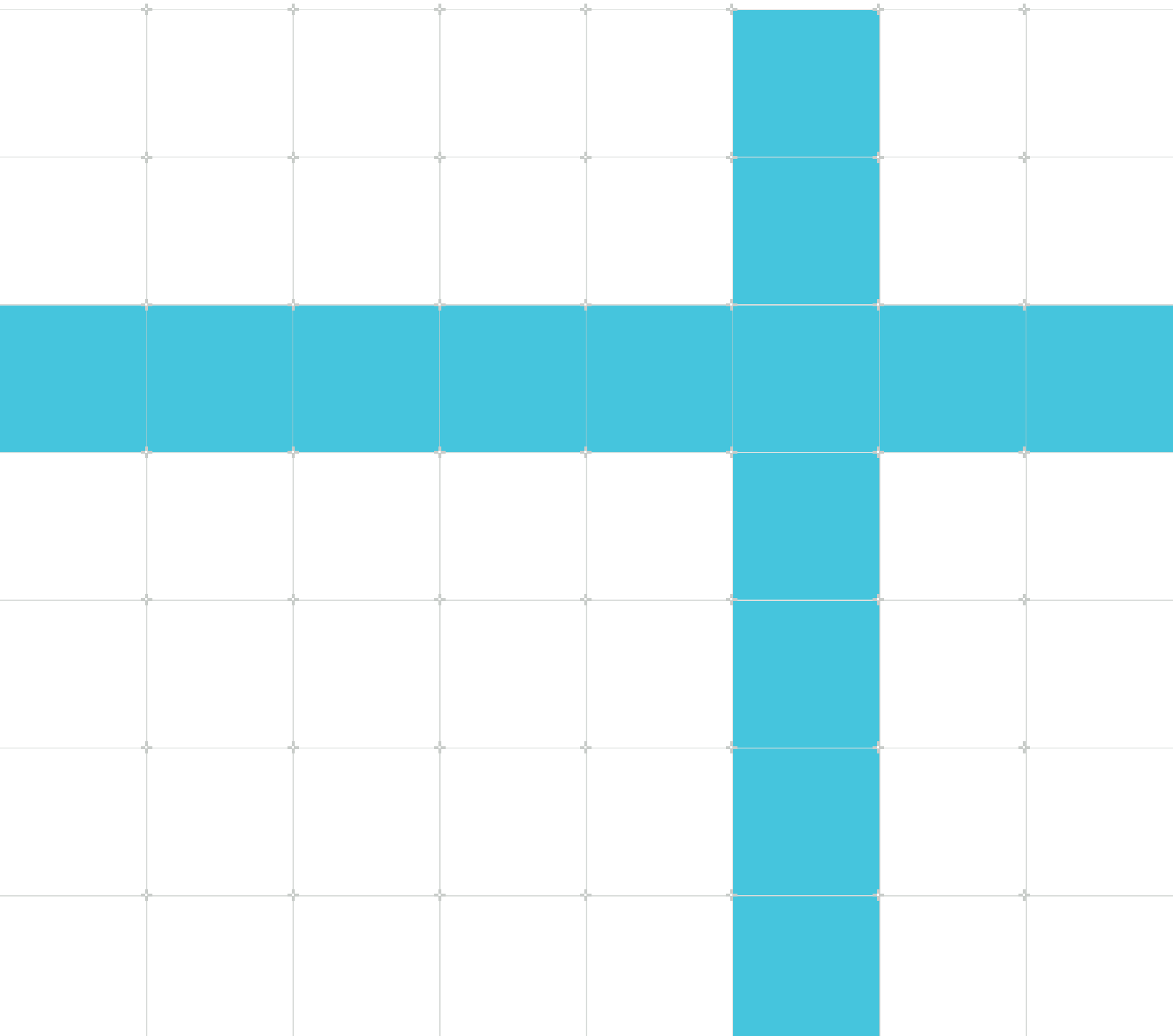
Arm® Cortex®-A510 Core Cryptographic Extension

Revision: r1p3

Technical Reference Manual

Non-Confidential

Issue 20
101606_0103_20_en



Arm® Cortex®-A510 Core Cryptographic Extension

Technical Reference Manual

Release Information

Document history

Issue	Date	Confidentiality	Change
0000-01	20 December 2019	Confidential	First beta release for r0p0
0000-06	17 July 2020	Confidential	First limited access release for r0p0
0001-08	23 October 2020	Confidential	First early access release for r0p1
0002-09	11 December 2020	Confidential	First early access release for r0p2
0100-13	14 May 2021	Confidential	First limited access release for r1p0
0101-17	10 September 2021	Confidential	First early access release for r1p1
0102-18	29 April 2022	Confidential	First release for r1p2
0102-19	28 June 2022	Non-Confidential	First Non-Confidental release for r1p2
0103-20	30 September 2022	Non-Confidential	First Non-Confidental release for r1p3

Proprietary Notice

DOCUMENT. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, patents, copyrights, trade secrets, or other rights.

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|version 21.0)

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 an Arm core.

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.

Convention	Use
italic	Citations.
bold	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 underline`	A permitted abbreviation for a command or option. You can enter the underlined text instead of the full command or option name.
`<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>`

Convention	Use
SMALL CAPITALS	Terms that have specific technical meanings as defined in the *Arm® Glossary*. For example, **IMPLEMENTATION DEFINED**, **IMPLEMENTATION SPECIFIC**, **UNKNOWN**, and **UNPREDICTABLE**.

Recommendations. Not following these recommendations might lead to system failure or damage.

Requirements for the system. Not following these requirements might result in system failure or damage.

Requirements for the system. Not following these requirements will result in system failure or damage.

An important piece of information that needs your attention.

A useful tip that might make it easier, better or faster to perform a task.

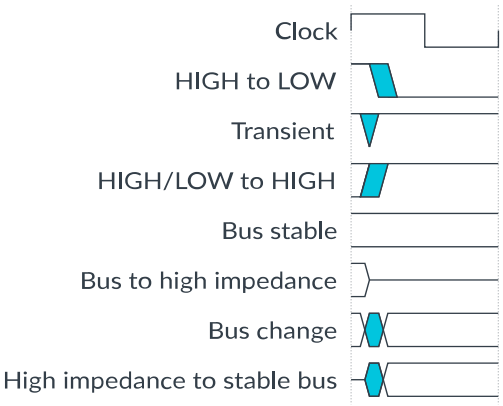
A reminder of something important that relates to the information you are reading.

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 Useful resources

This document contains information that is specific to this product. See the following resources for other useful information.

Access to Arm documents depends on their confidentiality:

- Non-Confidential documents are available at developer.arm.com/documentation. Each document link in the following tables goes to the online version of the document.
- Confidential documents are available to licensees only through the product package.

Arm product resources	Document ID	Confidentiality
Arm® Cortex®-A510 Core Configuration and Integration Manual	101605	Confidential
Arm® Cortex®-A510 Core Technical Reference Manual	101604	Non-Confidential
Cortex®-A510 Release Note	-	Confidential

Arm architecture and specifications	Document ID	Confidentiality
Arm® Architecture Reference Manual for A-profile architecture	DDI 0487	Non-Confidential

Non-Arm resources	Document ID	Organization
Advanced Encryption Standard (FIPS 197, November 2001)	FIPS 197	https://www.nist.gov/
Secure Hash Standard (SHS) (FIPS 180-4, August 2015)	FIPS 180-4	https://www.nist.gov/
Secure Hash Standard (SHS) (FIPS 202, August 2015)	FIPS 202	https://www.nist.gov/

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. Cryptographic extension support in the Cortex®-A510 core

The Cortex®-A510 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 for r0p0
r0p1	Further development and optimization of the product, including addition of the *TRace Buffer Extension* (TRBE)
r0p2	Maintenance release
r1p0	First release for r1p0 includes the following features: • Optional support for AArch32 Execution state • *Memory Tagging Extension* (MTE) asymmetric fault handling • Enhancement for *Privileged Access Never* (PAN) with Execute-only
r1p1	First release for r1p1 includes: • Support for asymmetric VPU datapath width across complexes at cluster level • *Power Performance and Area* (PPA) improvements and bug fixes
r1p2	First release for r1p2 includes: • Support for FEAT_ECBHB, Exploitative Control using Branch History Buffer information between exception levels
r1p2	First Non-Confidential release for r1p2 includes: • Change in confidentiality from confidential to non-confidential • Update product name
r1p3	First Non-Confidential release for r1p3 includes: • Imported autogenerated register bundle, created using DITAGenerator v1.3.4

Changes in functionality that have an impact on the documentation also appear in A.1 Revisions on page 21.

2.2 Disabling the Cryptographic Extension

Disabling of the Cryptographic Extension applies to all Cortex®-A510 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 indicates that the Cryptographic Extension is not implemented.

Related information

2.4 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0 on page 11

2.3 Cryptographic Extensions register summary

The Cortex®-A510 core has a single instruction identification register, ID_AA64ISAR0_EL1. Software can identify the cryptographic instructions that are implemented by reading this register. The Cortex®-A510 core also provides ID_AA64ZFR0_EL1 and ID_ISAR5_EL1 as part of the Cryptographic Extension.

The following table shows the registers for the Cortex®-A510 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_AA64ZFR0_EL1	AArch64	See 2.5 ID_AA64ZFR0_EL1, SVE Feature ID register 0 on page 15
ID_ISAR5_EL1	AArch64	See 2.6 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5 on page 17

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* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure 2-1: AArch64_id_aa64isar0_el1 bit assignments

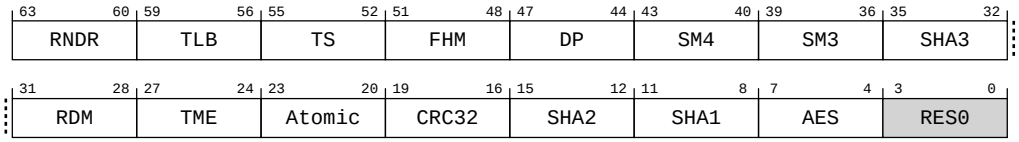


Table 2-3: ID_AA64ISAR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:60]	RNDR	Indicates support for Random Number instructions in AArch64 state. Defined values are: **0b0000** No Random Number instructions are implemented.	xxxx
[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.	xxxx
[55:52]	TS	Indicates support for flag manipulation instructions. Defined values are: **0b0010** CFINV, RMIF, SETF16, SETF8, AXFLAG, and XAFLAG instructions are implemented.	xxxx
[51:48]	FHM	Indicates support for FMLAL and FMLSL instructions. Defined values are: **0b0001** FMLAL and FMLSL instructions are implemented.	xxxx
[47:44]	DP	Indicates support for Dot Product instructions in AArch64 state. Defined values are: **0b0001** UDOT and SDOT instructions implemented.	xxxx

Bits	Name	Description	Reset
[43:40]	SM4	Indicates support for SM4 instructions in AArch64 state. Defined values are: **0b0000** No SM4 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SM4E and SM4EKEY instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[39:36]	SM3	Indicates support for SM3 instructions in AArch64 state. Defined values are: **0b0000** No SM3 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SM3SS1, SM3TT1A, SM3TT1B, SM3TT2A, SM3TT2B, SM3PARTW1, and SM3PARTW2 instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[35:32]	SHA3	Indicates support for SHA3 instructions in AArch64 state. Defined values are: **0b0000** No SHA3 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** EOR3, RAX1, XAR, and BCAX instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[31:28]	RDM	Indicates support for SQRDMLAH and SQRDMLSH instructions in AArch64 state. Defined values are: **0b0001** SQRDMLAH and SQRDMLSH instructions implemented.	xxxx
[27:24]	TME	Indicates support for TME instructions. Defined values are: **0b0000** TME instructions are not implemented.	xxxx
[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.	xxxx
[19:16]	CRC32	Indicates support for CRC32 instructions in AArch64 state. Defined values are: **0b0001** CRC32B, CRC32H, CRC32W, CRC32X, CRC32CB, CRC32CH, CRC32CW, and CRC32CX instructions implemented.	xxxx

Bits	Name	Description	Reset
[15:12]	SHA2	Indicates support for SHA2 instructions in AArch64 state. Defined values are: **0b0000** No SHA2 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0010** SHA256H, SHA256H2, SHA256SU0, SHA256SU1, SHA512H, SHA512H2, SHA512SU0, and SHA512SU1 instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	xxxx
[11:8]	SHA1	Indicates support for SHA1 instructions in AArch64 state. Defined values are: **0b0000** No SHA1 instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SHA1C, SHA1P, SHA1M, SHA1H, SHA1SU0, and SHA1SU1 instructions implemented. This value is reported when Cryptographic extensions are implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	xxxx
[7:4]	AES	Indicates support for AES instructions in AArch64 state. Defined values are: **0b0000** No AES instructions implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0010** AESE, AESD, AESMC, and AESIMC instructions are implemented plus PMULL/PMULL2 instructions operating on 64-bit data quantities. This value is reported when Cryptographic extensions are implemented and enabled. When the CRYPTO configuration parameter is true and the CRYPTODISABLE input is low at reset Cryptographic Extensions are implemented	xxxx
[3:0]	**RES0**	Reserved	**RES0**

Access

MRS <Xt>, ID_AA64ISAR0_EL1

op0	op1	CRn	CRm	op2
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_AA64ZFR0_EL1, SVE Feature ID register 0

Provides additional information about the implemented features of the AArch64 Scalable Vector Extension, when the AArch64-ID_AA64PFR0_EL1.SVE field is not zero.

For general information about the interpretation of the ID registers, see *Principles of the ID scheme for fields in ID registers* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations


Note

Prior to the introduction of the features described by this register, this register was unnamed and reserved, RES0 from EL1, EL2, and EL3.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx xxxx


Note

Where the reset reads xxxx, see individual bits

Bit descriptions

Figure 2-2: AArch64_id_aa64zfr0_el1 bit assignments

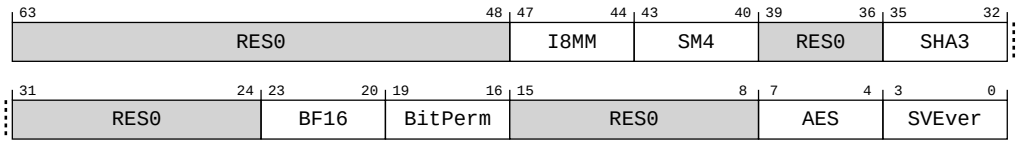


Table 2-5: ID_AA64ZFR0_EL1 bit descriptions

Bits	Name	Description	Reset
[63:48]	**RES0**	Reserved	**RES0**
[47:44]	I8MM	Indicates support for SVE Int8 matrix multiplication instructions. Defined values are: **0b0001** SMMLA, SUDOT, UMMLA, USMMLA, and USDOT instructions are implemented.	xxxx
[43:40]	SM4	Indicates support for SVE SM4 instructions. Defined values are: **0b0000** SVE2 SM4 instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SVE2 SM4E and SM4EKEY instructions are implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[39:36]	**RES0**	Reserved	**RES0**
[35:32]	SHA3	Indicates support for the SVE SHA3 instructions. Defined values are: **0b0000** SVE2 SHA-3 instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0001** SVE2 RAX1 instruction is implemented. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[31:24]	**RES0**	Reserved	**RES0**
[23:20]	BF16	Indicates support for SVE BFloat16 instructions. Defined values are: **0b0001** BFCVT, BFCVTNT, BFDOT, BFMLALB, BFMLALT, and BFMMLA instructions are implemented.	xxxx
[19:16]	BitPerm	Indicates support for SVE bit permute instructions. Defined values are: **0b0001** SVE BDEP, BEXT, and BGRP instructions are implemented.	xxxx
[15:8]	**RES0**	Reserved	**RES0**

Bits	Name	Description	Reset
[7:4]	AES	Indicates support for SVE AES instructions. Defined values are: **0b0000** SVE2-AES instructions are not implemented. This value is reported when Cryptographic extensions are not implemented or are disabled. **0b0010** SVE2 AESE, AESD, AESMC, and AESIMC instructions are implemented plus SVE2 PMULLB and PMULLT instructions with 64-bit source. This value is reported when Cryptographic extensions are implemented and enabled.	xxxx
[3:0]	SVEver	Indicates support for SVE version 2. Defined values are: **0b0001** SVE and the non-optional SVE2 instructions are implemented.	xxxx

Access

MRS <Xt>, ID_AA64ZFR0_EL1

op0	op1	CRn	CRm	op2
0b11	0b000	0b0000	0b0100	0b100

Accessibility

MRS <Xt>, ID_AA64ZFR0_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_AA64ZFR0_EL1;
elsif PSTATE.EL == EL2 then
    return ID_AA64ZFR0_EL1;
elsif PSTATE.EL == EL3 then
    return ID_AA64ZFR0_EL1;
```

2.6 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* in the *Arm® Architecture Reference Manual for A-profile architecture*.

Configurations

This register is available in all configurations.

Attributes

Width

64

Functional group

Identification registers

Access type

See bit descriptions

Reset value

When HaveAnyAArch32()

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Note

Where the reset reads xxxx, see individual bits

Bit descriptions

When HaveAnyAArch32()

Figure 2-3: AArch64_id_isar5_el1 bit assignments

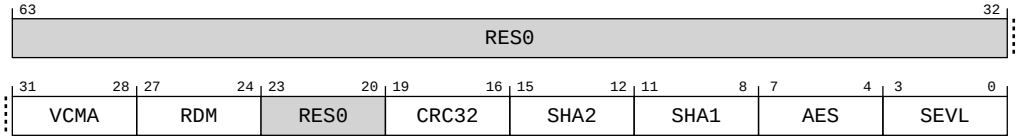


Table 2-7: ID_ISAR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:32]	**RES0**	Reserved	**RES0**
[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.	xxxx
[27:24]	RDM	Indicates whether the VQRDMLAH and VQRDMLSH instructions are implemented in AArch32 state. Defined values are: **0b0001** VQRDMLAH and VQRDMLSH instructions implemented.	xxxx

Bits	Name	Description	Reset
[23:20]	**RES0**	Reserved	**RES0**
[19:16]	CRC32	Indicates whether the CRC32 instructions are implemented in AArch32 state. **0b0001** CRC32B, CRC32H, CRC32W, CRC32CB, CRC32CH, and CRC32CW instructions implemented.	xxxx
[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.	xxxx
[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.	xxxx
[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.	xxxx
[3:0]	SEVL	Indicates whether the SEVL instruction is implemented in AArch32 state. **0b0001** SEVL is implemented as Send Event Local.	xxxx

Figure 2-4: AArch64_id_isar5_el1 bit assignments

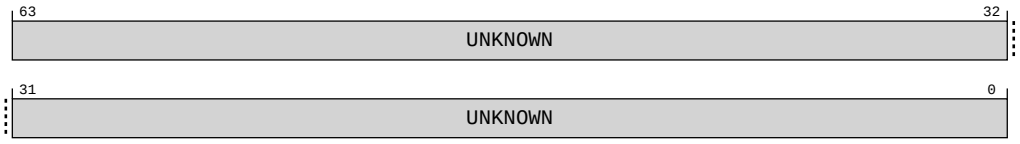


Table 2-8: ID_ISAR5_EL1 bit descriptions

Bits	Name	Description	Reset
[63:0]	**UNKNOWN**	Reserved	**UNKNOWN**

Access

MRS <Xt>, ID_ISAR5_EL1

op0	op1	CRn	CRm	op2
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.

Table A-1: Issue 0000-01

Change	Location
First Confidential beta release for r0p0	-

Table A-2: Differences between issue 0000-01 and issue 0000-06

Change	Location
First Confidential limited access release for r0p0	-
Updated register description	2.4 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0 on page 11
Added new section	2.5 ID_AA64ZFR0_EL1, SVE Feature ID register 0 on page 15

Table A-3: Differences between issue 0000-06 and issue 0001-08

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

Table A-4: Differences between issue 0001-08 and issue 0002-09

Change	Location
First Confidential early access release for r0p2	-
No technical changes	-

Table A-5: Differences between issue 0002-09 and issue 0100-13

Change	Location
First limited access release for r1p0	-
Minor clarifications to register description and accessibility description	2.4 ID_AA64ISAR0_EL1, AArch64 Instruction Set Attribute Register 0 on page 11
	2.5 ID_AA64ZFR0_EL1, SVE Feature ID register 0 on page 15

Table A-6: Differences between issue 0100-13 and issue 0101-17

Change	Location
First early access release for r1p1	-
No technical changes	-

Table A-7: Differences between issue 0101-17 and issue 0102-18

Change	Location
First release for r1p2	-
Added new register	2.6 ID_ISAR5_EL1, AArch32 Instruction Set Attribute Register 5 on page 17

Table A-8: Differences between issue 0102-18 and issue 0102-19

Change	Location
First Non-Confidential release for r1p2	-
Updated product name to Cortex-A510 core	Throughout the document
Updated confidentiality to non-confidential	Throughout the document

Table A-9: Differences between issue 0102-19 and issue 0103-20

Change	Location
First Non-Confidential release for r1p3	-
Updated product revision table	2.1 Product revisions on page 10
Imported autogenerated register bundle, created using DITAGenerator v1.3.4	Throughout the document
Updated useful resources tables	1.4 Useful resources on page 8