TrustZone® Address Space Controller (TZC-380)

Revision: r0p0

Technical Reference Manual

ARM®

ARM DDI 0431B (ID4/1/10)

TrustZone Address Space Controller (TZC-380)
Technical Reference Manual

Release Information

The *Change history* table lists the changes made to this book.

Change history

Date	Issue	Confidentiality	Change
10 September 2008	A	Confidential	First release for r0p0
19 March 2010	B	Non-Confidential	Second release for r0p0

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.

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
TrustZone Address Space Controller (TZC-380) Technical Reference Manual

List of Tables
TrustZone Address Space Controller (TZC-380) Technical Reference Manual

List of Figures
TrustZone Address Space Controller (TZC-380) Technical Reference Manual

Preface

This preface introduces the *TrustZone Address Space Controller (TZC-380) Technical Reference Manual* (TRM). It contains the following sections:

About this book

This is the TRM for the *TrustZone Address Space Controller (TZC-380)*.

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 programmers who are designing or programming a *System-on-Chip* (SoC) that uses the TZASC.

Using this book

This book is organized into the following chapters:

Chapter 1 *Introduction*

Read this for an introduction to the TZASC and its features.

Chapter 2 *Functional Description*

Read this for a description of the major interfaces and the functional operation of the TZASC.

Chapter 3 *Programmers Model*

Read this for a description of the memory map and registers.

Chapter 4 *Programmers Model for Test*

Read this for a description of the test registers.

Appendix A *Signal Descriptions*

Read this for a description of the input and output signals.

Appendix B *Revisions*

Read this for a description of the technical differences between consecutive revisions of this book.

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

Conventions

Conventions that this book can use are described in:

- *Typographical*
- *Timing diagrams* on page xi
- *Signals* on page xi.

Typographical

The typographical conventions are:

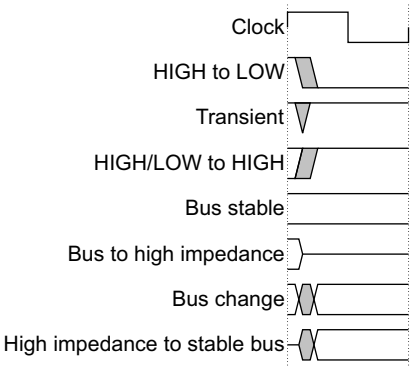
italic Highlights important notes, introduces special terminology, denotes internal cross-references, and citations.

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.
`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>`

Timing diagrams

The figure named *Key to timing diagram conventions* 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.



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.
Prefix a	Denotes global *Advanced eXtensible Interface* (AXI) signals.
Prefix ar	Denotes AXI read address channel signals.
Prefix aw	Denotes AXI write address channel signals.
Prefix b	Denotes AXI write response channel signals.

Prefix p	Denotes *Advanced Peripheral Bus* (APB) signals.
Prefix n	Denotes an active-LOW signal.
Prefix r	Denotes AXI read data channel signals.
Prefix w	Denotes AXI write data channel signals.

Additional reading

This section lists publications by ARM and by third parties.

See `http://infocenter.arm.com` for access to ARM documentation.

ARM publications

This book contains information that is specific to the TZASC. See the following documents for other relevant information:

- *TrustZone Address Space Controller (TZC-380) Implementation Guide* (ARM DII 0249)
- *TrustZone Address Space Controller (TZC-380) Integration Manual* (ARM DII 0250)
- *TrustZone Address Space Controller (TZC-380) Supplement to AMBA Designer (ADR-301) User Guide* (ARM DSU 0014)
- *AMBA Designer (ADR-301) User Guide* (ARM DUI 0333)
- *AMBA AXI Protocol Specification* (ARM IHI 0022)
- *AMBA 3 APB Protocol Specification* (ARM IHI 0024)
- *TrustZone Address Space Controller (TZC-380) Release Note*
- *AMBA Specification (Rev 2.0)* (ARM IHI 0011).

Other publications

This section lists relevant documents published by third parties:

- JEDEC Solid State Technology Association, *JEP106, Standard Manufacturer's Identification Code*, obtainable at `http://www.jedec.org`.

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 0431B
- 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 *TrustZone Address Space Controller (TZC-380)*. It contains the following sections:

1.1 About the TZASC

The TZASC is an *Advanced Microcontroller Bus Architecture* (AMBA) compliant *System-on-Chip* (SoC) peripheral. It is a high-performance, area-optimized address space controller with on-chip AMBA bus interfaces that conform to the AMBA *Advanced eXtensible Interface* (AXI) protocol and the AMBA *Advanced Peripheral Bus* (APB) protocol.

You can configure the TZASC to provide the optimum security address region control functions required for your intended application. See *Features of the TZASC* for a summary of the configurable features supported.

Figure 1-1 shows the interfaces that are available on the TZASC.

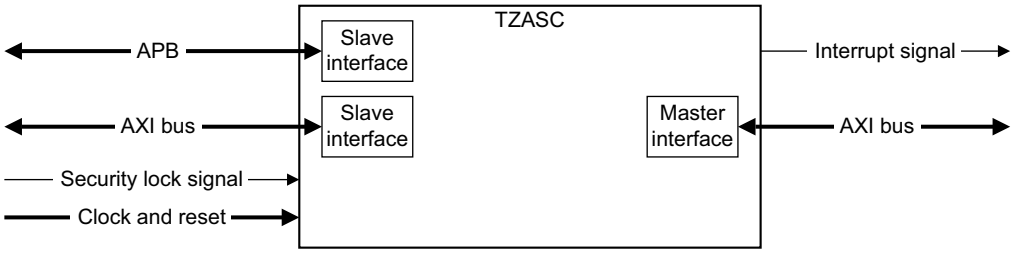


Figure 1-1 Interfaces on the TZASC

Figure 1-2 shows the TZASC in an example system.

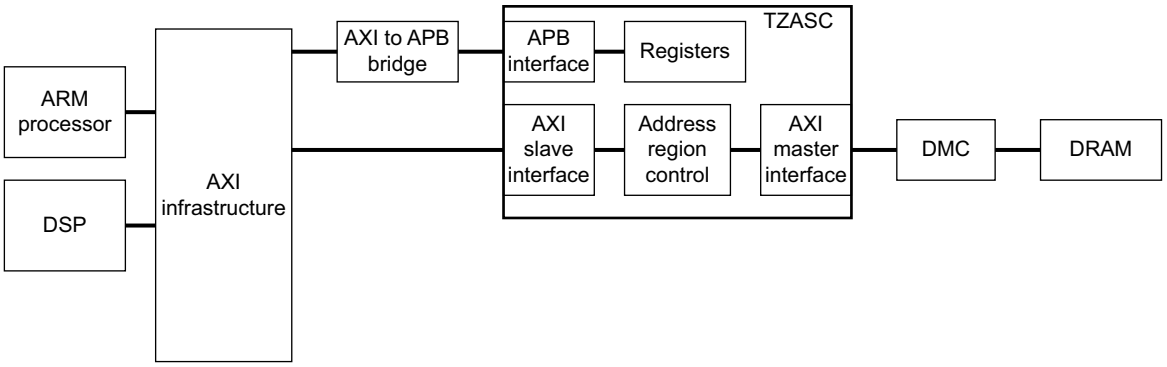


Figure 1-2 Example system

The TZASC example system contains:

- an AXI to APB bridge restricting APB traffic to the TZASC to secure only
- a TZASC
- AXI bus masters:
  - an ARM processor
  - a *Digital Signal Processor* (DSP).
- AXI infrastructure component
- a *Dynamic Memory Controller* (DMC).

1.1.1 Features of the TZASC

The TZASC provides the following features:

- enables you to program security access permissions for each address region
- permits the transfer of data between master and slave only if the security status of the AXI transaction matches the security settings of the memory region it addresses

- prevents write access to various registers after assertion of **secure_boot_lock**.

You can configure the:

- AXI address bus width to be 32-bit to 64-bit inclusive
- AXI data bus width to be 32-bit, 64-bit, 128-bit, or 256-bit
- AXI ID tag width to be 1-bit to 24-bit inclusive
- AXI **USER** bus width to be 0-bit to 32-bit inclusive
- number of address regions to be:
  - 2 regions
  - 4 regions
  - 8 regions
  - 16 regions.
- APB protocol version to be AMBA 2 or AMBA 3
- transaction tracking queue depth to be 1-16 inclusive
- timing close options:
  - no pipeline stages
  - forward direction timing isolation on slave interface and master interface AXI ports
  - complete timing isolation on slave interface and master interface AXI ports.

1.2 Product revisions

This section describes the differences in functionality between product revisions:

r0p0 First release.

Chapter 2
Functional Description

This chapter describes the TZASC operation. It contains the following sections:

2.1 Functional interfaces

The main interfaces of the TZASC are:

- *AXI bus interfaces*
- *APB slave interface* on page 2-3
- *Miscellaneous signals* on page 2-3
- *Clock and reset* on page 2-4.

2.1.1 AXI bus interfaces

The TZASC provides the following AXI bus interfaces:

- *AXI slave interface*
- *AXI master interface* on page 2-3.

Each AXI bus interface consists of the following AXI channels:

- Write Address (AW)
- Write Data (W)
- Write Response (B)
- Read Address (AR)
- Read Data (R).

See the *AMBA AXI Protocol v1.0 Specification* for information about the AXI protocol.

AXI slave interface

Table 2-1 shows the AXI slave interface attributes and their values.

Table 2-1 AXI slave interface attributes

Attribute[a]	Value
Read acceptance capability	Equals the configurable transaction tracker queue depth. If register slices are enabled on AR channel, the read acceptance capability is configurable transaction tracker queue depth plus one.
Write acceptance capability	Equals the configurable transaction tracker queue depth. If register slices are enabled on AW channel, the write acceptance capability is configurable transaction tracker queue depth plus one.
Combined acceptance capability	Equals the configurable transaction tracker queue depth. If register slices are enabled on AR and AW channels, the combined acceptance capability is configurable transaction tracker queue depth plus two.
Write interleave depth	1
Read data reordering depth	Equal to zero. The TZASC does not re-order read data. However the TZASC supports the re-ordering depth of the downstream slave.

a. See *Glossary* for a description of these AXI attributes.

AXI master interface

Table 2-2 shows the AXI master interface attributes and their values.

Table 2-2 AXI master interface attributes

Attribute[a]	Value
Combined issuing capability	Equals the transaction tracking queue depth that is configurable
Write issuing capability	Equals the transaction tracking queue depth that is configurable
Read issuing capability	Equals the transaction tracking queue depth that is configurable

a. See *Glossary* for a description of these AXI attributes.

2.1.2 APB slave interface

The APB slave interfaces provide access to the TZASC registers that enables you to program the system configuration parameters and obtain status information. See Chapter 3 *Programmers Model* for more information.

Note

The APB slave interface must only be accessible to processors in Secure state, otherwise it can compromise the security of the system.

You can configure the TZASC to support either the:

- AMBA 2 APB protocol
- AMBA 3 APB protocol.

See the *AMBA 2 APB Protocol v1.0 Specification* or *AMBA 3 APB Protocol v1.0 Specification* for more information.

Note

If you configure the TZASC to support the AMBA 2 APB protocol, it provides all the signals, except **pready** and **pslverr**.

2.1.3 Miscellaneous signals

There are two miscellaneous signals:

- **secure_boot_lock**
- **tzasc_int**.

Figure 2-1 shows the miscellaneous signals that the TZASC provides.

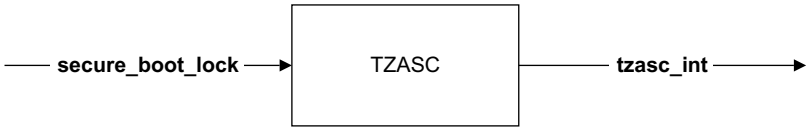


Figure 2-1 Miscellaneous signals

Asserting **secure_boot_lock** enhances the security of the TZASC. See *Preventing writes to registers and using secure_boot_lock* on page 2-12.

You can program the TZASC to assert **tzasc_int** when it denies an AXI master access to a region. See *Denied AXI transactions* on page 2-11.

2.1.4 Clock and reset

This section describes:

- *Clock*
- *Reset*
- *pclken*.

Clock

All configurations of the TZASC use a single clock input, **aclk**. See *Clock and reset signals* on page A-2.

Reset

The TZASC provides a single reset input, **aresetn**. See *Clock and reset signals* on page A-2.

pclken

Clock enable signal that enables the APB slave interface to operate at either:

- the **aclk** frequency
- a divided integer multiple of **aclk** that is synchronous to **aclk**.

Note

If you do not use **pclken**, you must tie it HIGH. This results in the APB slave interface being clocked directly by **aclk**.

2.2 Functional operation

TZASC is a systems IP that performs security checks on AXI accesses to memory, or off-chip peripheral. This supports configurable number of regions. Each region is programmable for size, base address, enable, and security parameters. Using the **secure_boot_lock**, the programmers view can be locked to prevent erroneous writes. See *Preventing writes to registers and using secure_boot_lock* on page 2-12. The IP provides programmability in reporting faults using AXI response channel, and interrupt.

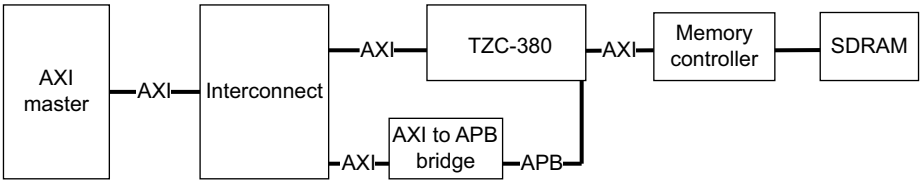


Figure 2-2 Functional operation of TZASC

The following sections describe:

- *Regions*
- *Priority* on page 2-6
- *Subregions* on page 2-6
- *Subregion disable* on page 2-7
- *Region security permissions* on page 2-8
- *Denied AXI transactions* on page 2-11
- *Speculative accesses* on page 2-12
- *Preventing writes to registers and using secure_boot_lock* on page 2-12
- *Using locked transaction sequences* on page 2-13
- *Using exclusive accesses* on page 2-13.

Note

The *TrustZone Address Space Controller (TZC-380) Supplement to AMBA Designer (ADR-301) User Guide* provides information about how to configure the controller.

2.2.1 Regions

A region is a contiguous area of address space. The TZASC provides each region with a programmable security permissions field. The security permissions value is used to enable the TZASC to either accept or deny a transaction access to that region. The transactions **arprots[2:0]** or **awprots[2:0]** signals are used to determine the security settings of that transaction.

The TZASC always provides two regions, region 0 and region 1, and you can configure it to provide additional regions. With the exception of region 0, the TZASC enables you to program the following operating parameters for each region:

- Region enable.
- Security permissions.
- Base address.
- Size. The minimum address size of a region is 32KB.
- Subregion disable. See *Subregions* on page 2-6.

Note

Region 0 is known as the background region because it occupies the total memory space. You can program the security permissions of region 0, but the following parameters are fixed:

Base address `0x0`

Size The `AXI_ADDRESS_MSB` configuration parameter controls the address range of the TZASC, and therefore the region size.

Subregion disable This feature is not available for region 0.

2.2.2 Priority

The priority of a region is fixed and is determined by the region number. Figure 2-3 shows how the priority of a region increases with the region number.

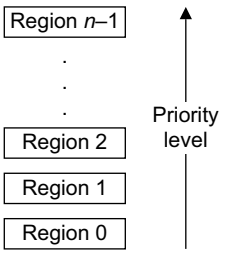


Figure 2-3 Region priority

When a transaction is received, its address is checked for a match with all the configured regions in turn. The order in which the regions are checked is determined by the priority level, the highest priority level is first. The first region that matches the transaction address match is used as the matching region. The matching regions security permission determines whether the transaction is permitted.

2.2.3 Subregions

The TZASC divides each region into eight equal-sized, non-overlapping subregions. Figure 2-4 on page 2-7 shows the subregions for an example region that is programmed to occupy an address span of 32KB.

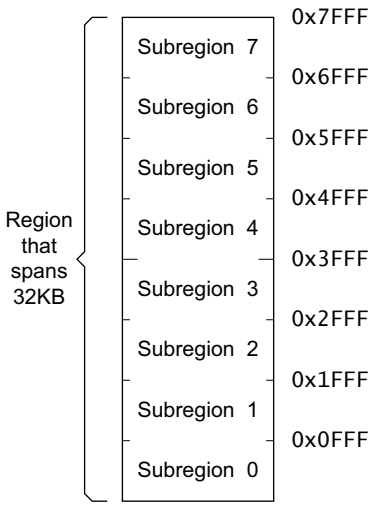


Figure 2-4 Subregion example

2.2.4 Subregion disable

With the exception of region 0, you can program the TZASC to disable any or all of the eight subregions that comprise a region. When a subregion is disabled, the security permissions for its address range are provided by the next highest priority region that overlaps the address range.

Example 2-1 Example configuration for subregion disable

Figure 2-5 on page 2-8 shows an example configuration that supports four regions, where:

- region 2 and region 3 are partially overlapped
- region 1 and region 3 are partially overlapped
- region 0 is overlapped with all regions.

With some subregions of region 1, region 2, and region 3 are disabled, and the resulting region permissions of the entire address space is shown in the Figure 2-5 on page 2-8.

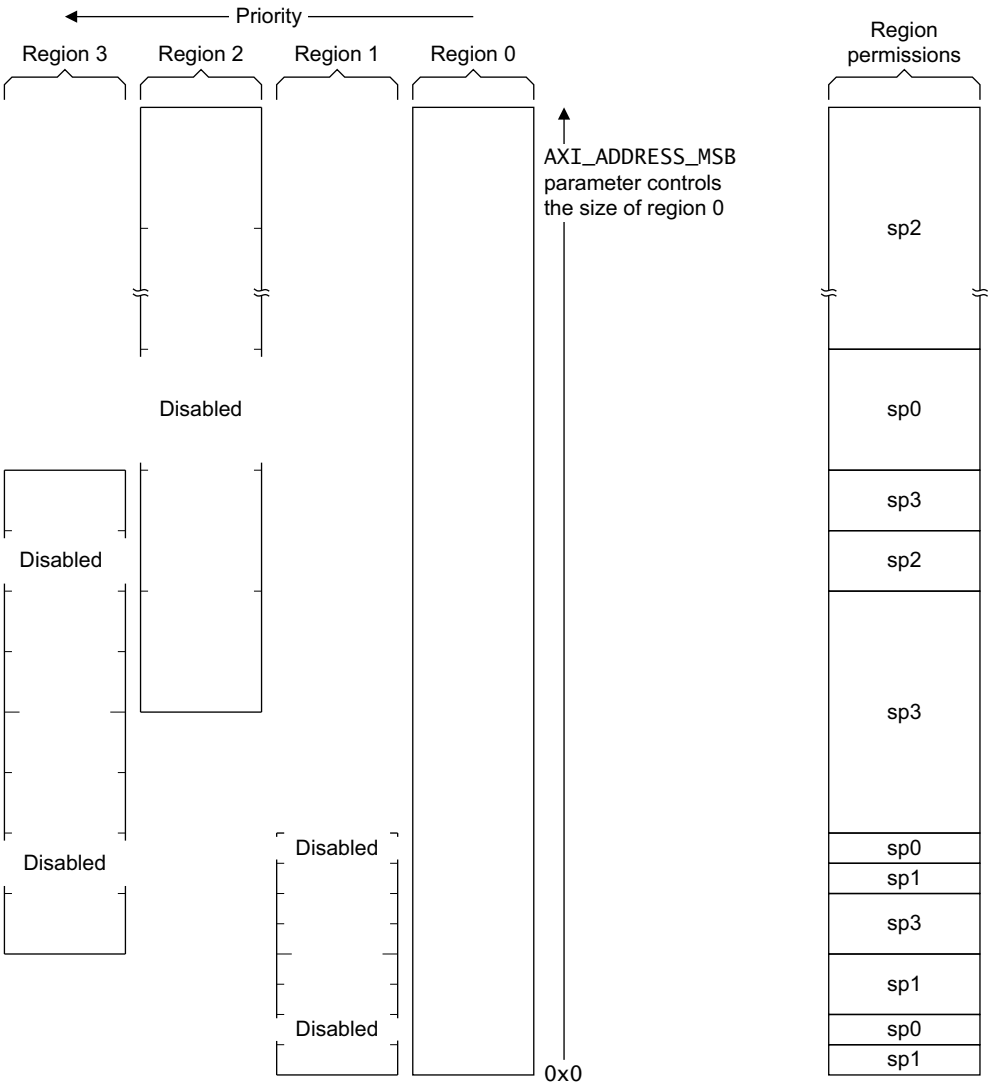


Figure 2-5 Subregion disable example

Note

In Figure 2-5:

- all subregions are enabled unless otherwise stated
- sp*n* represents the region permissions of region *n*.

2.2.5 Region security permissions

The TZASC enables you to program the security access permissions for any region that it is configured. A region is assigned a security permissions field, sp<n>, in its region_attributes_<n> Register that enables you to have complete control of the permissions for that region. See Chapter 4 *Programmers Model for Test*.

Security inversion

There are two modes of operation for the region security permissions, with or without security inversion.

By default, if you program a region to support non-secure accesses, the TZASC ensures that region must also support secure accesses. For example, if you program the region permissions for region 3 to be non-secure read only, the TZASC permits access to region 3 for secure reads and non-secure reads.

If you require that some regions are not accessible to masters in Secure state, but are accessible in Non-secure state, then you must enable security inversion.

See *Region security permissions* on page 2-8 and *Security Inversion Enable Register* on page 3-17 for more information.

Programming security permissions when security inversion is disabled

By default, security inversion is disabled and therefore the TZASC only permits you to program certain combinations of security permissions. These combinations ensure that a master in Secure state is not denied access to a region that is programmed to only accept non-secure accesses. Table 2-3 shows the possible security permissions when security inversion is disabled.

Table 2-3 Region security permissions when security inversion is disabled

	sp<n> field controls if the TZASC permits access for the following AXI transactions			
sp<n> field[a]	**Secure read**	**Secure write**	**Non-secure read**	**Non-secure write**
b0000	No	No	No	No
b0100	No	Yes	No	No
b0001, b0101	No	Yes	No	Yes
b1000	Yes	No	No	No
b0010, b1010	Yes	No	Yes	No
b1100	Yes	Yes	No	No
b1001, b1101	Yes	Yes	No	Yes
b0110, b1110	Yes	Yes	Yes	No
b0011, b0111, b1011, b1111	Yes	Yes	Yes	Yes

a. See *Region Attributes <n> Register* on page 3-20 for programming information.

Programming security permissions when security inversion is enabled

If you enable security inversion, the TZASC permits you to program any combination of security permissions as Table 2-4 shows.

Table 2-4 Region security permissions when security inversion is enabled

	sp<n> field controls if the TZASC permits access for the following AXI transactions			
sp<n> field[a]	**Secure read**	**Secure write**	**Non-secure read**	**Non-secure write**
b0000	No	No	No	No
b0001	No	No	No	Yes
b0010	No	No	Yes	No
b0011	No	No	Yes	Yes

Table 2-4 Region security permissions when security inversion is enabled (continued)

	sp<n> field controls if the TZASC permits access for the following AXI transactions			
sp<n> field[a]	**Secure read**	**Secure write**	**Non-secure read**	**Non-secure write**
b0100	No	Yes	No	No
b0101	No	Yes	No	Yes
b0110	No	Yes	Yes	No
b0111	No	Yes	Yes	Yes
b1000	Yes	No	No	No
b1001	Yes	No	No	Yes
b1010	Yes	No	Yes	No
b1011	Yes	No	Yes	Yes
b1100	Yes	Yes	No	No
b1101	Yes	Yes	No	Yes
b1110	Yes	Yes	Yes	No
b1111	Yes	Yes	Yes	Yes

a. See *Region Attributes <n> Register* on page 3-20 for programming information.

Table 2-5 shows a typical example of memory map along with the register programming. The TZASC is configured to have 16 regions.

Table 2-5 Typical example of memory map along with the register programming

Region	Region[a]	Lock	Starting address	Region size	Size field	sp[b]	Description
Region_0 (Default)	Enable	No	0x0	max	-	1100	Secure *Read Write* access (RW).
Region_1	Enable	No	0x0	64MB	b011001	1111	Non-secure *Read or Write access* (R/W), Secure R/W.
Region_2	Enable	No	0x0	16MB	b010111	1110	Non-secure *Read Only access* (RO), Secure RW for the normal world OS kernel.
Region_3	Enable	No	0x3D00000	512KB	b010010	1111	Regularly switched Non-secure, or Secure RW for a more complex shared memory buffers.
Region_4	Enable	No	0x3D80000	512KB	b010010	1100	Non-secure *No Access* (NA), Secure RW, a dedicated area for secure LCD Controller frame buffer.
Region_5[c]	Enable	No	0x80000000	32KB	b001110	1111	Non-secure RW, Secure RW for address range of general peripherals such as screen control, and keyboard hardware.
Region_6	Enable	Yes	0x3C00000	512KB	b010010	1011	Non-secure RW, Secure RO for streaming from the normal world to the secure world.
Region_7	Enable	Yes	0x3C80000	512KB	b010010	1110	Non-secure RO, Secure RW for streaming from the secure world to the normal world.

Table 2-5 Typical example of memory map along with the register programming (continued)

Region	Region[a]	Lock	Starting address	Region size	Size field	sp[b]	Description
Region_8	Enable	Yes	0x3E00000	512KB	b010010	1000	Non-secure NA, Secure RO for the secure world OS kernel.
Region_9	Enable	Yes	0x3E80000	512KB	b010010	1100	Non-secure NA, Secure RW for the secure world OS applications, heap, and stacks.
Region_10	Enable	Yes	0x3F00000	1MB	b010011	1100	Non-secure NA, Secure RW for Secure world OS applications, heap, and stacks.
Region_11[c]	Enable	Yes	0x80008000	32KB	b001110	1100	Non-secure NA, Secure RW for address range of secure peripherals such as *Random Number Generator* (RNG), and cryptography support hardware.
Region_12	Enable	Yes	0xF0000000	256MB	b011011	0011	Non-secure RW, Secure NA for FLASH holding normal world OS plus disk.
Region_13	Enable	Yes	0xF0000000	1MB	b010011	1100	Non-secure NA, Secure RW for FLASH for secure boot, secure world OS, secure configuration details.
Region_14	Disable	-	-	-	-	-	-
Region_15	Disable	-	-	-	-	-	-

a. Region can be either Enable or Disable.
b. *Security Permission* (sp).
c. In a more typical system, these devices would be protected by a *TrustZone Protection Controller* (BP147), and associated TrustZone aware *AXI to APB Bridges* (BP135).

Note

The implementers system design, and security requirements are taken into account for this example. And any actual software programming must depend on the system where TZASC is plugged.

2.2.6 Denied AXI transactions

If an AXI transaction has insufficient security privileges then for:

Reads The TZASC responds to the master by setting all bits of the read data bus, **rdatas[AXI_DATA_MSB:0]**, to zero.

Note

If the TZASC is programmed to perform speculative accesses, it discards the data that it receives on **rdatam[AXI_DATA_MSB:0]**.

Writes The TZASC prevents the transfer of data from the master to the slave by discarding the data that **wdatas[AXI_DATA_MSB:0]** contains. If you program the TZASC to perform speculative accesses, it modifies the transfer to the slave by setting all bits of the:

- write data bus, **wdatam[AXI_DATA_MSB:0]**, to zero
- write data strobe, **wstrbm[AXI_STRB_MSB:0]**, to zero.

Note

The action Register controls whether the TZASC signals to the master when a region permission failure occurs, and if so, the type of response it provides. See *Action Register* on page 3-8.

2.2.7 Speculative accesses

By default, the TZASC performs read or write speculative accesses that means it forwards an AXI transaction address to a slave, before it verifies that the AXI transaction is permitted to read address or write address respectively.

The TZASC only permits the transfer of data between its AXI bus interfaces, after verifying the access that the read or write access is permitted respectively. If the verification fails, then it prevents the transfer of data between the master and slave as *Denied AXI transactions* on page 2-11 describes.

You can disable speculative accesses by programming the speculation_control Register. See *Speculation Control Register* on page 3-16. When speculative accesses are disabled, the TZASC verifies the permissions of the access before it forwards the access to the slave. If the TZASC:

- Permits the access, it commences an AXI transaction to the slave, and it adds one clock latency.
- Denies the access, it prevents the transfer of data between the master and slave as *Denied AXI transactions* on page 2-11 describes. In this situation, the slave is unaware when the TZASC prevents the master from accessing the slave.

Note

Enabling speculative access is a potential security risk, if the device that is being protected reacts to this transaction. Most devices do not have to react to this level of access, and speculative access is much faster than validating the address before issuing the transaction.

2.2.8 Preventing writes to registers and using secure_boot_lock

By suitably programming lockdown Register, see *Lockdown Select Register* on page 3-9, and asserting **secure_boot_lock** signal makes the following registers read only:

- speculation_control Register. See *Speculation Control Register* on page 3-16.
- security_inversion_en Register. See *Security Inversion Enable Register* on page 3-17.
- lockdown_range Register. See *Lockdown Range Register* on page 3-9.

Locking down the region using lockdown_range and lockdown_select registers

By programming the lockdown_select, and lockdown_range registers, and asserting the **secure_boot_lock** signal, you can lockdown the behavior of the TZASC so that it prevents unintentional or erroneous write to the regions specified in the lockdown_range Register. However, read access to those regions is permitted:

- region_setup_low_<n> Register. See *Region Setup Low <n> Register* on page 3-18
- region_setup_high_<n> Register. See *Region Setup High <n> Register* on page 3-19
- region_attributes_<n> Register. See *Region Attributes <n> Register* on page 3-20.

The TZASC expects the **secure_boot_lock** signal to be asserted for at least one clock cycle. One clock after the **secure_boot_lock** is sampled HIGH by TZASC, then the registers mentioned in *Locking down the region using lockdown_range and lockdown_select registers* on page 2-12 cannot be written, unless the TZASC is reset by asserting **aresetn**.

2.2.9 Using locked transaction sequences

If a master performs locked transaction sequences, a transaction might stall, or an AXI protocol violation might occur when:

Transaction sequence crosses a 4KB boundary

If a locked transaction sequence crosses a 4KB boundary and the regions have different region permissions, the TZASC might prevent access to the second region and therefore the slave would not receive the latter part of the locked transaction sequence.

Note

The AXI protocol recommends that locked transaction sequences do not cross a 4KB address boundary.

Secure state change

During a locked transaction sequence, if a master changes the state of **arprots[1]** or **awprots[1]** and the region has different region permissions for Secure state and Non-secure state, the TZASC might deny a transaction and therefore the slave would not receive the latter part of the locked transaction sequence.

Reads and writes

During a locked transaction sequence, if a master performs reads and writes to a region, depending on the region permissions, the TZASC might deny a transaction and therefore the slave would not receive the latter part of the locked transaction sequence.

2.2.10 Using exclusive accesses

If a master performs exclusive accesses to an address region, you must program the TZASC to permit read and write accesses to that address region, for the expected settings of **arprots[1]** and **awprots[1]**, otherwise the read or write transaction might fail.

2.3 Constraints of use

The TZASC has the following considerations relating to change in programmers view on an active system:

- When changing the setting of a TZASC region,
  - The current accepted AXI transaction, if it falls into that region, would act according to the previous settings for that region.
  - Any other outstanding AXI transactions, that falls into that region, would effect by the new settings for that region.
- Given little ability to predict that the mentioned AXI transactions would effect, it is obviously desirable that there are no outstanding AXI transactions when a regions setting are changed.
  - In simple systems this can potentially be achieved by the core not accessing the given region during the period of the cores transition between security states. Even in these cases, the status of cached data and instructions needs to be considered.
  - In more complicated systems the code that changes the TZASC region settings must have to inform other AXI bus masters to desist or complete acting on that region before performing the region setting changes. After having such an action acknowledged the code must also have to instigate a suitable delay before then acting.

An example of this can be an LCD controller dealing with a frame buffer that is switching between a Normal world and Secure world use.

Note

There is no direct mechanism to ascertain if there are any outstanding AXI transactions, and so the designer must use their system knowledge to apply reasonable mechanisms.

It is recommended that any DECERR, or TZASC interrupt handler is designed to expect, and potentially ignore events generated under these circumstances.

Chapter 3
Programmers Model

This chapter describes the TZASC registers, and provides information for programming the device. It contains the following sections:

3.1 About the programmers model

The following information applies to the TZASC registers:

- The base address of the TZASC is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these location can result in Unpredictable behavior of the TZASC.
- Unless otherwise stated in the accompanying text:
  - do not modify undefined register bits
  - ignore undefined register bits on reads
  - all register bits are reset to a logic 0 by a system or power-on reset.
- For programming the registers, the TZASC supports data in word-invariant endianness.
- The Type column in Table 3-1 on page 3-5 describes the access types as follows:
  RW Read and write.
  RO Read only.
  WO Write only.
- System designers must ensure that only processors in Secure state can access the registers, otherwise it can compromise the security of the system.

Note

See *Constraints of use* on page 2-14 for more information about considerations relating to change in programmers view on an active system.

3.1.1 Register map

The register map of the TZASC spans a 4KB region, as Figure 3-1 shows.

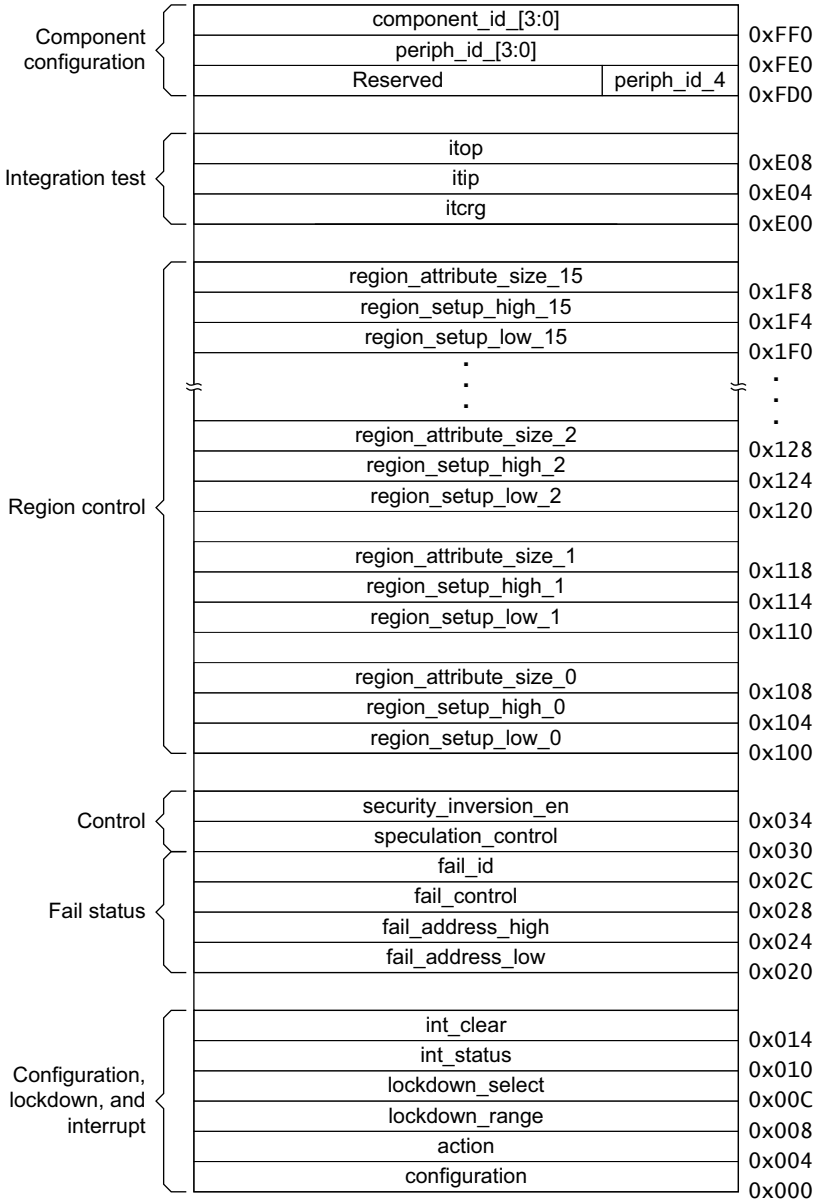


Figure 3-1 Register map

In Figure 3-1, the register map consists of the following regions:

Configuration, lockdown, and interrupt

Use these registers to determine the global configuration of the TZASC, and control its operating state.

Fail status These registers provide information about an access that failed because of insufficient permissions.

Control Use these registers to enable the TZASC to perform security inversion or speculative accesses.

Region control

Use these registers to control the operating state of each region.

Integration test

Use these registers when testing the integration of the TZASC in a *System-on-Chip* (SoC). See Chapter 4 *Programmers Model for Test* for more information.

Component configuration

These registers enable the identification of system components by software.

3.2 Register descriptions

This section describes the registers that the TZASC provides. Table 3-1 lists the registers.

Table 3-1 Register summary

Offset	Name	Type	Reset	Width	Description
0x000	configuration	RO	-[a]	32	*Configuration Register* on page 3-6
0x004	action	RW	0x00000001	32	*Action Register* on page 3-8
0x008	lockdown_range	RW[b]	0x00000000	32	*Lockdown Range Register* on page 3-9
0x00C	lockdown_select	RW[c]	0x00000000	32	*Lockdown Select Register* on page 3-9
0x010	int_status	RO	0x00000000	2	*Interrupt Status Register* on page 3-11
0x014	int_clear	WO	0x00000000	32	*Interrupt Clear Register* on page 3-12
0x018-0x01C	-	-	-	-	Reserved
0x020	fail_address_low	RO	0x00000000	32	*Fail Address Low Register* on page 3-12
0x024	fail_address_high	RO	0x00000000	32	*Fail Address High Register* on page 3-13
0x028	fail_control	RO	0x00000000	32	*Fail Control Register* on page 3-14
0x02C	fail_id	RO	0x00000000	-[d]	*Fail ID Register* on page 3-15
0x030	speculation_control	RW[b]	0x00000000	32	*Speculation Control Register* on page 3-16
0x034	security_inversion_en	RW[b]	0x00000000	32	*Security Inversion Enable Register* on page 3-17
0x038-0x0FC	-	-	-	-	Reserved
0x100	region_setup_low_0	RW[e]	0x00000000	32	*Region Setup Low <n> Register* on page 3-18
0x110	region_setup_low_1				
0x120	region_setup_low_2[f]				
.	.				
.	.				
.	.				
0x1F0	region_setup_low_15[f]				
0x104	region_setup_high_0	RW[e]	0x00000000	32	*Region Setup High <n> Register* on page 3-19
0x114	region_setup_high_1				
0x124	region_setup_high_2[f]				
.	.				
.	.				
.	.				
0x1F4	region_setup_high_15[f]				

Table 3-1 Register summary (continued)

Offset	Name	Type	Reset	Width	Description
0x108	region_attributes_0	RW	0xc0000000	32	*Region Attributes <n> Register* on page 3-20
0x118	region_attributes_1		0x0000001c		
0x128	region_attributes_2[f]				
.	.				
.	.				
.	.				
0x1F8	region_attributes_15[f]				
0x1nC[g]	-	-	-	-	Reserved
0x200-0xDFC	-	-	-	-	Reserved
0xE00	itcrg	See Chapter 4 *Programmers Model for Test* for information about these registers			
0xE04	itip				
0xE08	itop				
0xE0C-0xEFC	-	-	-	-	Reserved
0xFD0	periph_id_4	RO	0x00000004	8	*Peripheral Identification Registers* on page 3-23
0xFE0-0xFEC	periph_id_[3:0]	RO	0x000BB380[h]	8	
0xFF0-0xFFC	component_id_[3:0]	RO	0xB105F00D	8	*Component Identification Registers* on page 3-25

a. The reset value depends on the configuration of the TZASC.
b. Access type can become RO depending on **secure_boot_lock** and the value of the lockdown_select Register. See *Lockdown Select Register* on page 3-9.
c. Access type becomes RO if **secure_boot_lock** goes HIGH.
d. Dependant on configuration, range from 31-0.
e. Access type is RW for all regions, except region 0 is RO.
f. The configuration of the TZASC controls the number of regions, and therefore, the region_<…>_2 to region_<…>_15 registers that are available.
g. For values of *n* from 0x0 to 0xF inclusive.
h. The reset value depends on the revision of the TZASC. See *Peripheral Identification Register 2* on page 3-24.

3.2.1 Configuration Register

The configuration Register characteristics are:

Purpose Provides information about the configuration of the TZASC.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-2 shows the configuration Register bit assignments.

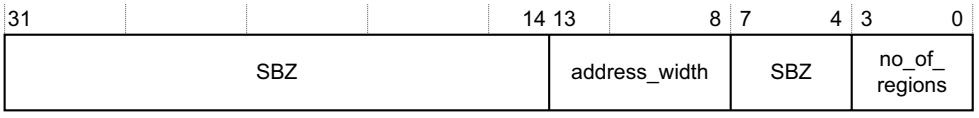


Figure 3-2 configuration Register bit assignments

Table 3-2 shows the configuration Register bit assignments.

Table 3-2 configuration Register bit assignments

Bits	Name	Function
[31:14]	-	Reserved, *Should be Zero* (SBZ).
[13:8]	address_width	Returns the width of the AXI address bus. Read as: b000000-b011110 = reserved b011111 = 32-bit b100000 = 33-bit b100001 = 34-bit . . . b111110 = 63-bit b111111 = 64-bit.
[7:4]	-	Reserved, SBZ.
[3:0]	no_of_regions	Returns the number of regions that the TZASC provides: b0000 = reserved b0001 = 2 regions b0010 = 3 regions b0011 = 4 regions . . . b1111 = 16 regions.

3.2.2 Action Register

The action Register characteristics are:

Purpose Controls the response signaling behavior of the TZASC to region permission failures.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-3 shows the action Register bit assignments.

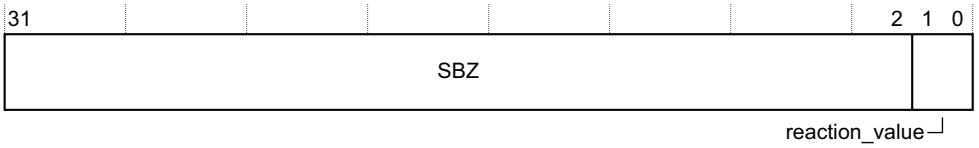


Figure 3-3 action Register bit assignments

Table 3-3 shows the action Register bit assignments.

Table 3-3 action Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved, SBZ
[1:0]	reaction_value	Controls how the TZASC uses the **bresps[1:0]**, **rresps[1:0]**, and **tzasc_int** signals when a region permission failure occurs: b00 = sets **tzasc_int** LOW and issues an OKAY response b01 = sets **tzasc_int** LOW and issues a DECERR response b10 = sets **tzasc_int** HIGH and issues an OKAY response b11 = sets **tzasc_int** HIGH and issues a DECERR response.

3.2.3 Lockdown Range Register

The lockdown_range Register characteristics are:

Purpose Controls the range of regions that are locked down.

Usage constraints The lockdown_select Register can restrict the access type of this register to RO. See *Lockdown Select Register*.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-4 shows the lockdown_range Register bit assignments.

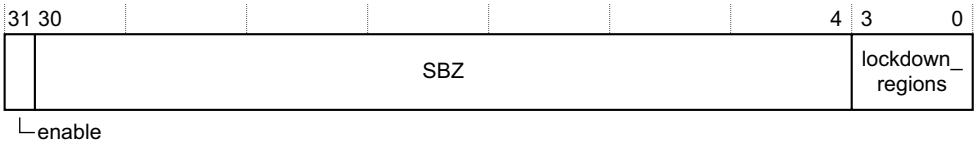


Figure 3-4 lockdown_range Register bit assignments

Table 3-4 shows the lockdown_range Register bit assignments.

Table 3-4 lockdown_range Register bit assignments

Bits	Name	Function
[31]	enable	When set to 1, it enables the lockdown_regions field to control the regions that are to be locked
[30:4]	-	Reserved, SBZ
[3:0]	lockdown_regions	Controls the number of regions to lockdown when the enable bit is set to 1: b0000 = region *no_of_regions*[a]–1 is locked b0001 = region *no_of_regions*–1 to region *no_of_regions*–2 are locked b0010 = region *no_of_regions*–1 to region *no_of_regions*–3 are locked b0011 = region *no_of_regions*–1 to region *no_of_regions*–4 are locked . . . b1111 = region *no_of_regions*–1 to region *no_of_regions*–16 are locked.

a. *no_of_regions* is the value of the no_of_regions field in the configuration Register. See *Configuration Register* on page 3-6.

3.2.4 Lockdown Select Register

The lockdown_select Register characteristics are:

Purpose Controls whether the TZASC permits write accesses to the following registers:

- *Lockdown Range Register*
- *Speculation Control Register* on page 3-16
- *Security Inversion Enable Register* on page 3-17.

Usage constraints After **aresetn** goes HIGH, the TZASC only permits write access to this register, if **secure_boot_lock** remains LOW. When **secure_boot_lock** is HIGH for one **aclk** period, or more then the TZASC ignores writes to this register.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-5 shows the lockdown_select Register bit assignments.

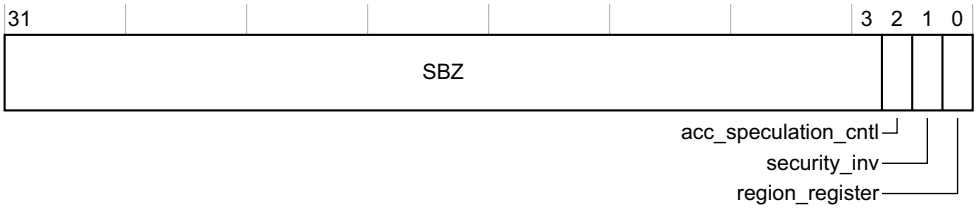


Figure 3-5 lockdown_select Register bit assignments

Table 3-5 shows the lockdown_select Register bit assignments.

Table 3-5 lockdown_select Register bit assignments

Bits	Name	Function
[31:3]	-	Reserved, SBZ.
[2]	acc_speculation_cntl	Modifies the access type of the speculation_control Register: 0 = no effect. speculation_control Register remains RW. 1 = speculation_control Register is RO. See *Speculation Control Register* on page 3-16 for more information.
[1]	security_inv	Modifies the access type of the security_inversion_en Register: 0 = no effect. security_inversion_en Register remains RW. 1 = security_inversion_en Register is RO. See *Security Inversion Enable Register* on page 3-17 for more information.
[0]	region_register	Modifies the access type of the lockdown_range Register: 0 = no effect. lockdown_range Register remains RW. 1 = lockdown_range Register is RO. See *Lockdown Range Register* on page 3-9 for more information.

3.2.5 Interrupt Status Register

The int_status Register characteristics are:

Purpose Returns the status of the interrupt.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-6 shows the int_status Register bit assignments.

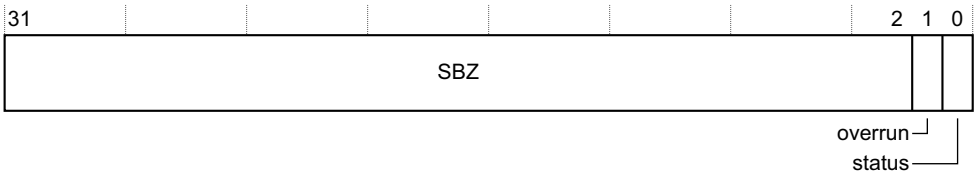


Figure 3-6 int_status Register bit assignments

Table 3-6 shows the int_status Register bit assignments.

Table 3-6 int_status Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved, SBZ
[1]	overrun	When set to 1, it indicates the occurrence of two or more region permission failures since the interrupt was last cleared
[0]	status	Returns the status of the interrupt: 0 = interrupt is inactive 1 = interrupt is active.

3.2.6 Interrupt Clear Register

The int_clear Register characteristics are:

Purpose Clears the interrupt.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Writing any value to the int_clear Register sets the:
- status bit to 0 in the int_status Register
- overrun bit to 0 in the int_status Register.

See *Interrupt Status Register* on page 3-11.

3.2.7 Fail Address Low Register

The fail_address_low Register characteristics are:

Purpose Returns the address, the lower 32-bits, of the first access that failed a region permission, after the interrupt was cleared.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-7 shows the fail_address_low Register bit assignments.

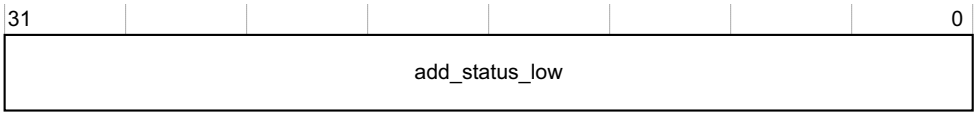


Figure 3-7 fail_address_low Register bit assignments

Table 3-7 shows the fail_address_low Register bit assignments.

Table 3-7 fail_address_low Register bit assignments

Bits	Name	Function
[31:0]	add_status_low	Returns the AXI address bits [31:0] of the first access to fail a region permission check after the interrupt was cleared.

3.2.8 Fail Address High Register

The fail_address_high Register characteristics are:

Purpose Returns the address, the upper 32-bits, of the first access that failed a region permission, after the interrupt was cleared.

Usage constraints There are no usage constraints.

Configurations Only available when the TZASC has an AXI address width of greater than 32 bits.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-8 shows the fail_address_high Register bit assignments.

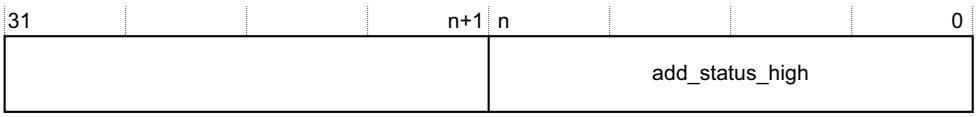


Figure 3-8 fail_address_high Register bit assignments

Table 3-8 shows the fail_address_high Register bit assignments.

Table 3-8 fail_address_high Register bit assignments

Bits	Name	Function
[31:n+1][a]	-	Reserved, SBZ
[n:0][a]	add_status_high	Returns the address bits [AXI_ADDRESS_MSB:32] of the first access to fail a region permission check after the interrupt was cleared

a. The value of n = AXI_ADDRESS_MSB–32.

3.2.9 Fail Control Register

The fail_control Register characteristics are:

Purpose Returns the control status information of the first access that failed a region permission, after the interrupt was cleared.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-9 shows the fail_control Register bit assignments.

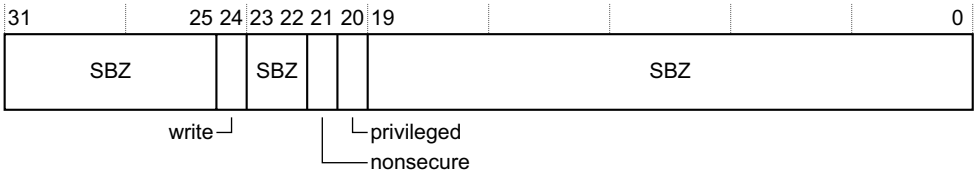


Figure 3-9 fail_control Register bit assignments

Table 3-9 shows the fail_control Register bit assignments.

Table 3-9 fail_control Register bit assignments

Bits	Name	Function
[31:25]	-	Reserved, SBZ.
[24]	write	This bit indicates whether the first access to fail a region permission check was a write or read as: 0 = read access 1 = write access.
[23:22]	-	Reserved, SBZ.
[21]	nonsecure	After clearing the interrupt status, this bit indicates whether the first access to fail a region permission check was non-secure. Read as: 0 = secure access 1 = non-secure access.
[20]	privileged	After clearing the interrupt status, this bit indicates whether the first access to fail a region permission check was privileged. Read as: 0 = unprivileged access 1 = privileged access.
[19:0]	-	Reserved, SBZ.

3.2.10 Fail ID Register

The fail_id Register characteristics are:

Purpose Returns the master AXI ID of the first access that failed a region permission, after the interrupt was cleared.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-10 shows the fail_id Register bit assignments.

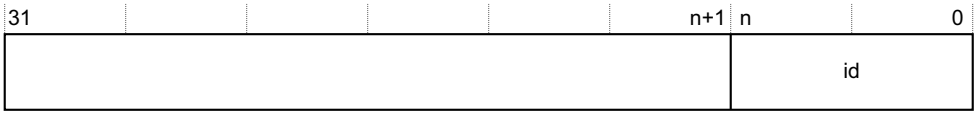


Figure 3-10 fail_id Register bit assignments

Table 3-10 shows the fail_id Register bit assignments.

Table 3-10 fail_id Register bit assignments

Bits	Name	Function
[31:n+1][a]	-	Reserved, SBZ
[n:0][a]	id	Returns the master AXI ID of the first access to fail a region permission check after the interrupt was cleared

a. The value of n = AID_WIDTH-1.

3.2.11 Speculation Control Register

The speculation_control Register characteristics are:

Purpose Controls the read access speculation and write access speculation.

Usage constraints The lockdown_select Register can restrict the access type of this register to RO. See *Lockdown Select Register* on page 3-9.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-11 shows the speculation_control Register bit assignments.

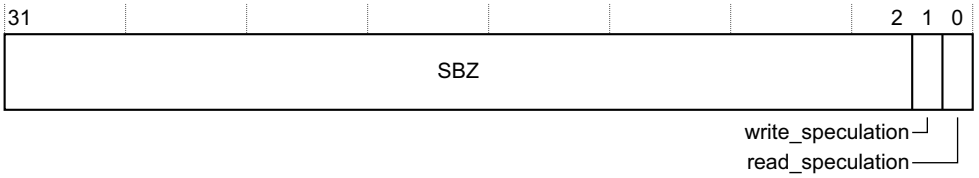


Figure 3-11 speculation_control Register bit assignments

Table 3-11 shows the speculation_control Register bit assignments.

Table 3-11 speculation_control Register bit assignments

Bits	Name	Function
[31:2]	-	Reserved, SBZ.
[1]	write_speculation	Controls the write access speculation: 0 = write access speculation is enabled. This is the default. 1 = write access speculation is disabled.
[0]	read_speculation	Controls the read access speculation: 0 = read access speculation is enabled. This is the default. 1 = read access speculation is disabled.

3.2.12 Security Inversion Enable Register

The security_inversion_en Register characteristics are:

Purpose Controls whether the TZASC enables security inversion to occur.

Usage constraints The lockdown_select Register can restrict the access type of this register to RO. See *Lockdown Select Register* on page 3-9.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-12 shows the security_inversion_en Register bit assignments.

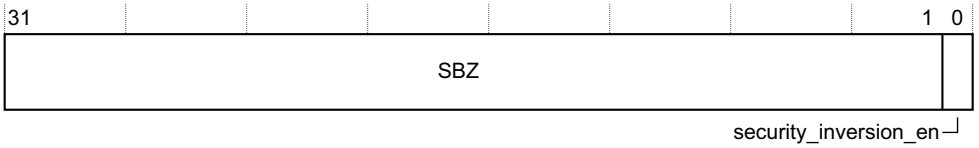


Figure 3-12 security_inversion_en Register bit assignments

Table 3-12 shows the security_inversion_en Register bit assignments.

Table 3-12 security_inversion_en Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved, SBZ.
[0]	security_inversion_en	Controls whether the TZASC permits security inversion to occur: 0 = security inversion is not permitted. This is the default. 1 = security inversion is permitted. This enables a region to be accessible to masters in Non-secure state but not accessible to masters in Secure state.

See *Security inversion* on page 2-8 for more information.

3.2.13 Region Setup Low <n> Register

The region_setup_low_<n> Register characteristics are:

Purpose None for region 0, for other regions it controls the base address [31:15] of region <n>.

Note

Base address [14:0] is always zero because the TZASC does not permit the region size to be less than 32KB.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-13 shows the region_setup_low_<n> Register bit assignments.

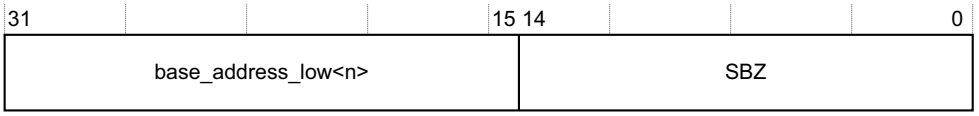


Figure 3-13 region_setup_low_<n> Register bit assignments

Table 3-13 shows the region_setup_low_<n> Register bit assignments.

Table 3-13 region_setup_low_<n> Register bit assignments

Bits	Name	Function
[31:15]	base_address_low<n>	Controls the base address [31:15] of region <n>[a]. The TZASC only permits a region to start at address `0x0`, or at a multiple of its region size. For example, if the size of a region is 512MB, and it is not at address `0x0`, the only valid settings for this field are: b00100000000000000 b01000000000000000 b01100000000000000 b10000000000000000 b10100000000000000 b11000000000000000 b11100000000000000. If you attempt to set an inappropriate base address for the size of the region, the TZASC ignores certain bits depending on the region size. See Table 3-16 on page 3-21 for more information.
[14:0]	-	Reserved, SBZ.

a. For region 0, this field is *Read Only* (RO). The TZASC sets the base address of region 0 to `0x0`.

3.2.14 Region Setup High <n> Register

The region_setup_high_<n> Register characteristics are:

Purpose None for region 0, for other regions it controls the base address [63:32] of region <n>.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-14 shows the region_setup_high_<n> Register bit assignments.

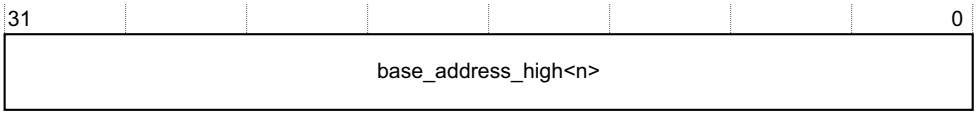


Figure 3-14 region_setup_high_<n> Register bit assignments

Table 3-14 shows the region_setup_high_<n> Register bit assignments.

Table 3-14 region_setup_high_<n> Register bit assignments

Bits	Name	Function
[31:0]	base_address_high<n>	Controls the base address [63:32] of region <n>[a]. The TZASC only permits a region to start at address `0x0`, or at a multiple of its region size. If you program a region size to be 8GB or more, then the TZASC might ignore certain bits depending on the region size. See Table 3-16 on page 3-21 for more information.

a. For region 0, this field is RO. The TZASC sets the base address of region 0 to `0x0`.

3.2.15 Region Attributes <n> Register

The region_attributes_<n> Register characteristics are:

Purpose Controls the permissions for region 0. For all other regions it controls the permissions, region size, subregion disable, and region enable.

Usage constraints There are no usage constraints.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 3-1 on page 3-5.

Figure 3-15 shows the region_attributes_<n> Register bit assignments.

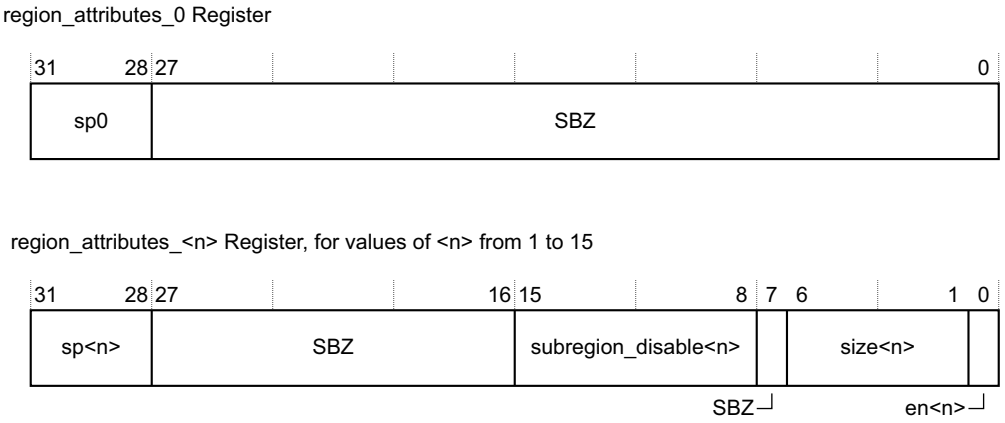


Figure 3-15 region_attributes_<n> Register bit assignments

Table 3-15 shows the region_attributes_<n> Register bit assignments.

Table 3-15 region_attributes_<n> Register bit assignments

Bits	Name	Function
[31:28]	sp<n>	Permissions setting for region <n>. If an AXI transaction occurs to region *n*, the value in the sp<n> field controls whether the TZASC permits the transaction to proceed. For more information, see: • *Programming security permissions when security inversion is disabled* on page 2-9 • *Programming security permissions when security inversion is enabled* on page 2-9.
[27:16]	-	Reserved, SBZ.
[15:8]	subregion_disable<n> [a]	Regions are split into eight equal-sized sub-regions, and each bit enables the corresponding subregion to be disabled: **Bit [15] = 1** Subregion 7 is disabled. **Bit [14] = 1** Subregion 6 is disabled. **Bit [13] = 1** Subregion 5 is disabled. **Bit [12] = 1** Subregion 4 is disabled. **Bit [11] = 1** Subregion 3 is disabled. **Bit [10] = 1** Subregion 2 is disabled. **Bit [9] = 1** Subregion 1 is disabled. **Bit [8] = 1** Subregion 0 is disabled. For more information, see *Subregions* on page 2-6 and *Subregion disable* on page 2-7.

Table 3-15 region_attributes_<n> Register bit assignments (continued)

Bits	Name	Function
[7]	-	Reserved, SBZ.
[6:1]	size<n>[a]	Size of region <n>. See Table 3-16. **Note** The AXI address width, that is AXI_ADDRESS_MSB+1, controls the upper limit value of this field.
[0]	en<n>[a]	Enable for region <n>: 0 = region <n> is disabled 1 = region <n> is enabled.

a. For region 0, this field is reserved.

Table 3-16 shows how the size<n> field controls the region size and what constraints, if any, the TZASC applies to the base address to ensure that a region starts on the boundary of the region size.

Table 3-16 Region size

size<n> field	Size of region <n>	Base address[a] constraints
b000000-b001101	Reserved	-
b001110	32KB	-
b001111	64KB	Bit [15] must be zero
b010000	128KB	Bits [16:15] must be zero
b010001	256KB	Bits [17:15] must be zero
b010010	512KB	Bits [18:15] must be zero
b010011	1MB	Bits [19:15] must be zero
b010100	2MB	Bits [20:15] must be zero
b010101	4MB	Bits [21:15] must be zero
b010110	8MB	Bits [22:15] must be zero
b010111	16MB	Bits [23:15] must be zero
b011000	32MB	Bits [24:15] must be zero
b011001	64MB	Bits [25:15] must be zero
b011010	128MB	Bits [26:15] must be zero
b011011	256MB	Bits [27:15] must be zero
b011100	512MB	Bits [28:15] must be zero
b011101	1GB	Bits [29:15] must be zero
b011110	2GB	Bits [30:15] must be zero
b011111	4GB	Bits [31:15] must be zero
b100000	8GB	Bits [32:15] must be zero

Table 3-16 Region size (continued)

size<n> field	Size of region <n>	Base address[a] constraints
b100001	16GB	Bits [33:15] must be zero
b100010	32GB	Bits [34:15] must be zero
b100011	64GB	Bits [35:15] must be zero
b100100	128GB	Bits [36:15] must be zero
b100101	256GB	Bits [37:15] must be zero
b100110	512GB	Bits [38:15] must be zero
b100111	1TB	Bits [39:15] must be zero
b101000	2TB	Bits [40:15] must be zero
b101001	4TB	Bits [41:15] must be zero
b101010	8TB	Bits [42:15] must be zero
b101011	16TB	Bits [43:15] must be zero
b101100	32TB	Bits [44:15] must be zero
b101101	64TB	Bits [45:15] must be zero
b101110	128TB	Bits [46:15] must be zero
b101111	256TB	Bits [47:15] must be zero
b110000	512TB	Bits [48:15] must be zero
b110001	1PB	Bits [49:15] must be zero
b110010	2PB	Bits [50:15] must be zero
b110011	4PB	Bits [51:15] must be zero
b110100	8PB	Bits [52:15] must be zero
b110101	16PB	Bits [53:15] must be zero
b110110	32PB	Bits [54:15] must be zero
b110111	64PB	Bits [55:15] must be zero
b111000	128PB	Bits [56:15] must be zero
b111001	256PB	Bits [57:15] must be zero
b111010	512PB	Bits [58:15] must be zero
b111011	1EB	Bits [59:15] must be zero
b111100	2EB	Bits [60:15] must be zero
b111101	4EB	Bits [61:15] must be zero
b111110	8EB	Bits [62:15] must be zero
b111111	16EB	Bits [63:15] must be zero

a. The region_setup_low_<n> Register and region_setup_high_<n> Register contain the base address. See Table 3-13 on page 3-18 and Table 3-14 on page 3-19.

3.2.16 Peripheral Identification Registers

The periph_id_[4:0] Registers provide information about the configuration of the peripheral. Table 3-1 on page 3-5 shows the address base offset, reset value, and access type for these registers.

Each register provides eight bits of data, but because some fields span across two adjacent periph_id registers, the following sections describe them:

- *periph_id_[3:0] register group*
- *Peripheral Identification Register 4* on page 3-25.

periph_id_[3:0] register group

Figure 3-16 shows the periph_id_[3:0] register group bit assignments.

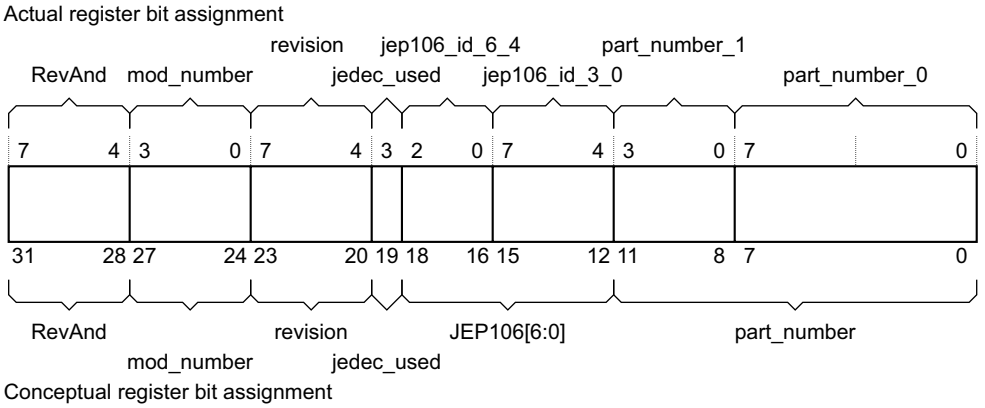


Figure 3-16 periph_id_[3:0] Register bit assignments

Table 3-17 shows the periph_id_[3:0] register group bit assignments.

Table 3-17 periph_id_[3:0] Register bit assignments

Bits	Name	Function
[31:28]	RevAnd	Identifies the manufacturer revision number.
[27:24]	mod_number	Identifies data that is relevant to the ARM partner.
[23:20]	revision	Identifies the revision of the TZASC.
[19]	jedec_used	Identifies whether the TZASC uses the JEP106 manufacturer's identity code.
[18:12]	JEP106[6:0]	Identifies the designer. This is set to b0111011, to indicate that ARM designed the peripheral.
[11:0]	part_number	Identifies the peripheral. The part number for the TZASC is `0x380`.

The following subsections describe the periph_id_[4:0] registers:

- *Peripheral Identification Register 0* on page 3-24
- *Peripheral Identification Register 1* on page 3-24
- *Peripheral Identification Register 2* on page 3-24
- *Peripheral Identification Register 3* on page 3-25
- *Peripheral Identification Register 4* on page 3-25.

Peripheral Identification Register 0

The periph_id_0 Register is hard-coded and the fields in the register control the reset value. Table 3-18 shows the register bit assignments.

Table 3-18 periph_id_0 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	part_number_0	These bits read back as `0x80`

Peripheral Identification Register 1

The periph_id_1 Register is hard-coded and the fields in the register control the reset value. Table 3-19 shows the register bit assignments.

Table 3-19 periph_id_1 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined.
[7:4]	jep106_id_3_0	JEP106 identity code [3:0]. See the *JEP106, Standard Manufacturer's Identification Code*. These bits read back as `0xB` because ARM is the designer of the peripheral.
[3:0]	part_number_1	These bits read back as `0x3`.

Peripheral Identification Register 2

The periph_id_2 Register is hard-coded and the fields in the register control the reset value. Table 3-20 shows the register bit assignments.

Table 3-20 periph_id_2 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined.
[7:4]	revision	Identifies the revision of the TZASC. For revision r0p0, this field is set to `0x0`.
[3]	jedec_used	This indicates that the TZASC uses a manufacturer's identity code that was allocated by JEDEC according to JEP106. This bit always reads back as `0x1`.
[2:0]	jep106_id_6_4	JEP106 identity code [6:4]. See the *JEP106, Standard Manufacturer's Identification Code*. These bits read back as b011 because ARM is the designer of the peripheral.

Peripheral Identification Register 3

The periph_id_3 Register is hard-coded and the fields in the register control the reset value. Table 3-21 shows the register bit assignments.

Table 3-21 periph_id_3 Register bit assignments

Bit	Name	Description
[31:8]	-	Undefined.
[7:4]	RevAnd	The top-level RTL provides four AND gates that are tied-off to provide an output value of `0x0`. When silicon is available, if metal fixes are necessary, the manufacturer can modify the tie-offs to indicate that a revision of the silicon has occurred.
[3:0]	mod_number	You can update this field by modifying the RTL of the TZASC. ARM sets this to `0x0`.

Peripheral Identification Register 4

The periph_id_4 Register is hard-coded and the fields in the register control the reset value. Figure 3-17 shows the periph_id_4 Register bit assignments.

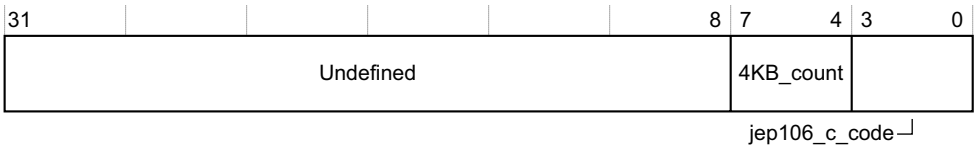


Figure 3-17 periph_id_4 Register bit assignments

Table 3-22 shows the register bit assignments.

Table 3-22 periph_id_4 Register bit assignments

Bits	Name	Function
[31:8]	-	Undefined.
[7:4]	4KB_count	The number of 4KB address blocks you require, to access the registers, expressed in powers of 2. These bits read back as `0x0`.
[3:0]	jep106_c_code	The JEP106 continuation code value represents how many `0x7F` continuation characters occur in the manufacturer's identity code. See *JEP106, Standard Manufacturer's Identification Code*. These bits read back as `0x4`.

3.2.17 Component Identification Registers

The component_id_[3:0] Registers are four, 8-bit wide registers, that can conceptually be treated as a single register that holds a 32-bit Component ID value. You can use the register for automatic BIOS configuration. The component_id Register is set to `0xB105F00D`. Table 3-1 on page 3-5 shows the address base offset, reset value, and access type for these registers.

Table 3-23 shows the register bit assignments.

Table 3-23 component_id Register bit assignments

component_id_0-3 register				
Bits	**Reset value**	**Register**	**Bits**	**Function**
-	-	component_id_3	[31:8]	Read undefined
[31:24]	0xB1	component_id_3	[7:0]	These bits read back as 0xB1
-	-	component_id_2	[31:8]	Read undefined
[23:16]	0x05	component_id_2	[7:0]	These bits read back as 0x05
-	-	component_id_1	[31:8]	Read undefined
[15:8]	0xF0	component_id_1	[7:0]	These bits read back as 0xF0
-	-	component_id_0	[31:8]	Read undefined
[7:0]	0x0D	component_id_0	[7:0]	These bits read back as 0x0D

Figure 3-18 shows the register bit assignments.

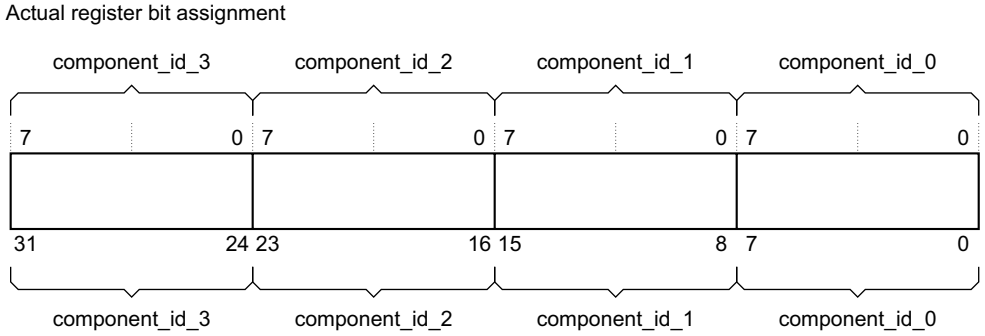


Figure 3-18 Component ID Register bit assignments

The following subsections describe the component_id Registers:

- *Component Identification Register 0* on page 3-27
- *Component Identification Register 1* on page 3-27
- *Component Identification Register 2* on page 3-27
- *Component Identification Register 3* on page 3-27.

Note

You cannot read these registers in the Reset state.

Component Identification Register 0

The component_id_0 Register is hard-coded and the fields in the register control the reset value. Table 3-24 shows the register bit assignments.

Table 3-24 component_id_0 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	component_id_0	These bits read back as `0x0D`

Component Identification Register 1

The component_id_1 Register is hard-coded and the fields in the register control the reset value. Table 3-25 shows the register bit assignments.

Table 3-25 component_id_1 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	component_id_1	These bits read back as `0xF0`

Component Identification Register 2

The component_id_2 Register is hard-coded and the fields in the register control the reset value. Table 3-26 shows the register bit assignments.

Table 3-26 component_id_2 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	component_id_2	These bits read back as `0x5`

Component Identification Register 3

The component_id_3 Register is hard-coded and the fields in the register control the reset value. Table 3-27 shows the register bit assignments.

Table 3-27 component_id_3 Register bit assignments

Bits	Name	Function
[31:8]	-	Reserved, read undefined
[7:0]	component_id_3	These bits read back as `0xB1`

Chapter 4
Programmers Model for Test

This chapter describes the registers for functional verification and integration testing. It contains the following sections:

4.1 About the programmers model for test

The following information applies to the TZASC registers:

- The base address of the TZASC is not fixed, and can be different for any particular system implementation. The offset of each register from the base address is fixed.
- Do not attempt to access reserved or unused address locations. Attempting to access these locations can result in unpredictable behavior of the TZASC.
- Unless otherwise stated in the accompanying text:
  - do not modify undefined register bits
  - ignore undefined register bits on reads
  - all register bits are reset to a logic 0 by a system or power-on reset.
- For programming the registers, the TZASC supports data in word-invariant endianness.
- The Type column in Table 4-1 on page 4-3 describes the access types as follows:
  RW Read and write.
  RO Read only.
  WO Write only.
- System designers must ensure that only processors in Secure state can access the registers, otherwise it can compromise the security of the system.

4.2 Integration test registers

Table 4-1 lists the integration test registers that the TZASC provides.

Table 4-1 Test register summary

Offset	Name	Type	Reset	Width	Description
0xE00	itcrg	RW	0x00000000	32	*Integration Test Control Register* on page 4-4
0xE04	itip	RO	0x00000000	32	*Integration Test Input Register* on page 4-5
0xE08	itop	RW	0x00000000	32	*Integration Test Output Register* on page 4-6
0xE0C-0xEFC	-	-	-	-	Reserved

4.2.1 Integration Test Control Register

The itcrg Register characteristics are:

Purpose Enables the integration test logic.

Usage constraints Use this in integration test mode.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 4-1 on page 4-3.

Figure 4-1 shows the itcrg Register bit assignments.

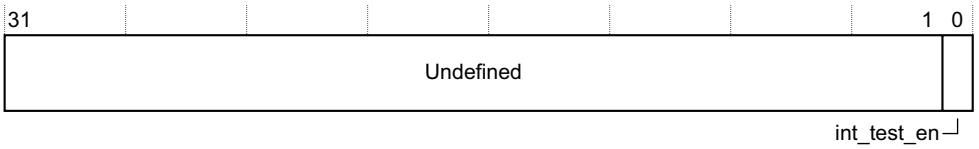


Figure 4-1 itcrg Register bit assignments

Table 4-2 shows the itcrg Register bit assignments.

Table 4-2 itcrg Register bit assignments

Bits	Name	Function
[31:1]	-	Undefined. Write as zero.
[0]	int_test_en	Controls the enabling of, or provides the status of, the integration test logic: 0 = integration test logic is disabled 1 = integration test logic is enabled.

4.2.2 Integration Test Input Register

The itip Register characteristics are:

Purpose Enables a processor to read the status of **secure_boot_lock**.

Usage constraints Integration test logic must be enabled otherwise reads return `0x0`. See *Integration Test Control Register* on page 4-4 for information about enabling the integration test logic.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 4-1 on page 4-3.

Figure 4-2 shows the itip Register bit assignments.

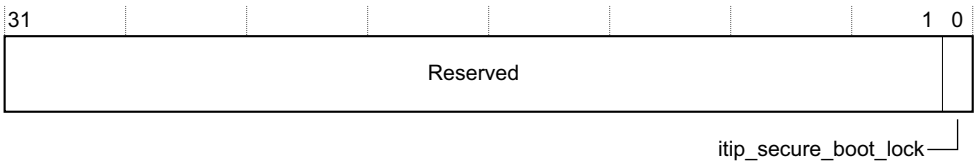


Figure 4-2 itip Register bit assignments

Table 4-3 shows the itip Register bit assignments.

Table 4-3 itip Register bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	itip_secure_boot_lock	Returns the status of **secure_boot_lock**: 0 = **secure_boot_lock** is LOW 1 = **secure_boot_lock** is HIGH.

4.2.3 Integration Test Output Register

The itop Register characteristics are:

Purpose Enables a processor to set the status of **tzasc_int** in integration test mode.

Usage constraints Integration test logic must be enabled otherwise it ignores writes and reads return `0x0`. See *Integration Test Control Register* on page 4-4 for information about enabling the integration test logic.

Configurations Available in all configurations of the TZASC.

Attributes See the register summary in Table 4-1 on page 4-3.

Figure 4-3 shows the itop Register bit assignments.

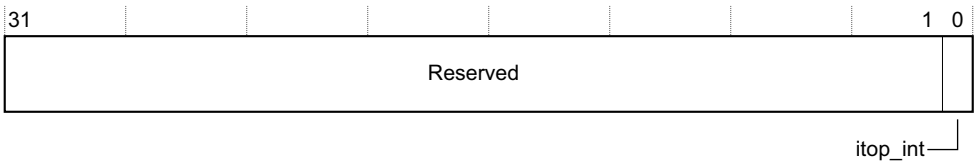


Figure 4-3 itop Register bit assignments

Table 4-4 shows the itop Register bit assignments.

Table 4-4 itop Register bit assignments

Bits	Name	Function
[31:1]	-	Read undefined. Write as zero.
[0]	itop_int	Set or reset the value of **tzasc_int** port by writing 1 or 0 into **itop_int** bit. If you read, the written value can be read back. 0 = **tzasc_int** is LOW 1 = **tzasc_int** is HIGH.

Appendix A
Signal Descriptions

This appendix describes the signals that the TZASC provides. It contains the following sections:

A.1 Clock and reset signals

Table A-1 shows the clock and reset signals.

Table A-1 Clock and reset signals

Signal	Type	Source	Description
aclk	Input	Clock source	Clock for the TZASC.
pclken	Input	Clock generator	Clock enable signal that enables the APB slave interface to operate at either: • the **aclk** frequency • a divided integer multiple of **aclk** that is aligned to **aclk**. **Note** If you do not use **pclken**, you must tie it HIGH. This results in the APB slave interface being clocked directly by **aclk**.
aresetn	Input	Reset source	Reset for the TZASC. This signal is active LOW.

A.2 AXI signals

The TZASC provides the following AXI bus interfaces:

- *AXI slave interface signals*
- *AXI master interface signals* on page A-5.

A.2.1 AXI slave interface signals

The following sections describe the AXI slave interface signals:

- *Write address (AXI-AW) channel signals*
- *Write data (AXI-W) channel signals* on page A-4
- *Write response (AXI-B) channel signals* on page A-4
- *Read address (AXI-AR) channel signals* on page A-4
- *Read data (AXI-R) channel signals* on page A-5.

Write address (AXI-AW) channel signals

Table A-2 lists the AXI write address signals for the AXI slave interface.

Table A-2 AXI-AW signals for the AXI slave interface

Signal	AMBA equivalent[a]
awaddrs[AXI_ADDRESS_MSB:0][b]	**AWADDR[31:0]**
awbursts[1:0]	**AWBURST[1:0]**
awcaches[3:0]	**AWCACHE[3:0]**
awids[AID_WIDTH–1:0][b]	**AWID[3:0]**
awlens[3:0]	**AWLEN[3:0]**
awlocks[1:0]	**AWLOCK[1:0]**
awprots[2:0]	**AWPROT[2:0]**
awreadys	**AWREADY**
awsizes[2:0]	**AWSIZE[2:0]**
awusers[AWUSER_WIDTH–1:0][bc]	-
awvalids	**AWVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

Write data (AXI-W) channel signals

Table A-3 lists the AXI write data signals for the AXI slave interface.

Table A-3 AXI-W signals for the AXI slave interface

Signal	AMBA equivalent[a]
wdatas[AXI_DATA_MSB:0][b]	**WDATA[31:0]**
wids[AID_WIDTH–1:0][b]	**WID[3:0]**
wlasts	**WLAST**
wreadys	**WREADY**
wstrbs[AXI_STRB_MSB:0][b]	**WSTRB[3:0]**
wusers[WUSER_WIDTH–1:0][bc]	-
wvalids	**WVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

Write response (AXI-B) channel signals

Table A-4 lists the AXI write response signals for the AXI slave interface.

Table A-4 AXI-B signals for the AXI slave interface

Signal	AMBA equivalent[a]
bids[AID_WIDTH–1:0][b]	**BID[3:0]**
breadys	**BREADY**
bresps[1:0]	**BRESP[1:0]**
busers[BUSER_WIDTH–1:0][bc]	-
bvalids	**BVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

Read address (AXI-AR) channel signals

Table A-5 lists the AXI read address signals for the AXI slave interface.

Table A-5 AXI-AR signals for the AXI slave interface

Signal	AMBA equivalent[a]
araddrs[AXI_ADDRESS_MSB:0][b]	**ARADDR[31:0]**
arbursts[1:0]	**ARBURST[1:0]**
arcaches[3:0]	**ARCACHE[3:0]**

Table A-5 AXI-AR signals for the AXI slave interface (continued)

Signal	AMBA equivalent[a]
arids[AID_WIDTH–1:0][b]	**ARID[3:0]**
arlens[3:0]	**ARLEN[3:0]**
arlocks[1:0]	**ARLOCK[1:0]**
arprots[2:0]	**ARPROT[2:0]**
arreadys	**ARREADY**
arsizes[2:0]	**ARSIZE[2:0]**
arusers[ARUSER_WIDTH–1:0][bc]	-
arvalids	**ARVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

Read data (AXI-R) channel signals

Table A-6 lists the AXI read data signals for the AXI slave interface.

Table A-6 AXI-R signals for the AXI slave interface

Signal	AMBA equivalent[a]
rdatas[AXI_DATA_MSB:0][b]	**RDATA[31:0]**
rids[AID_WIDTH–1:0][b]	**RID[3:0]**
rlasts	**RLAST**
rreadys	**RREADY**
rresps[1:0]	**RRESP[3:0]**
rusers[RUSER_WIDTH–1:0][bc]	-
rvalids	**RVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

A.2.2 AXI master interface signals

The following sections describe the AXI master interface signals:

- *Write address (AXI-AW) channel signals* on page A-6
- *Write data (AXI-W) channel signals* on page A-6
- *Write response (AXI-B) channel signals* on page A-7
- *Read address (AXI-AR) channel signals* on page A-7
- *Read data (AXI-R) channel signals* on page A-8.

Write address (AXI-AW) channel signals

Table A-7 lists the AXI write address signals for the AXI master interface.

Table A-7 AXI-AW signals for the AXI master interface

Signal	AMBA equivalent[a]
awaddrm[AXI_ADDRESS_MSB:0][b]	**AWADDR[31:0]**
awburstm[1:0]	**AWBURST[1:0]**
awcachem[3:0]	**AWCACHE[3:0]**
awidm[AID_WIDTH–1:0][b]	**AWID[3:0]**
awlenm[3:0]	**AWLEN[3:0]**
awlockm[1:0]	**AWLOCK[1:0]**
awprotm[2:0]	**AWPROT[2:0]**
awreadym	**AWREADY**
awsizem[2:0]	**AWSIZE[2:0]**
awuserm[AWUSER_WIDTH–1:0][bc]	-
awvalidm	**AWVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

Write data (AXI-W) channel signals

Table A-8 lists the AXI write data signals for the AXI master interface.

Table A-8 AXI-W signals for the AXI master interface

Signal	AMBA equivalent[a]
wdatam[AXI_DATA_MSB:0][b]	**WDATA[31:0]**
widm[AID_WIDTH–1:0][b]	**WID[3:0]**
wlastm	**WLAST**
wreadym	**WREADY**
wstrbm[AXI_STRB_MSB:0][b]	**WSTRB[3:0]**
wuserm[WUSER_WIDTH–1:0][bc]	-
wvalidm	**WVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

Write response (AXI-B) channel signals

Table A-9 lists the AXI write response signals for the AXI master interface.

Table A-9 AXI-B signals for the AXI master interface

Signal	AMBA equivalent[a]
bidm[AID_WIDTH–1:0][b]	**BID[3:0]**
breadym	**BREADY**
brespm[1:0]	**BRESP[1:0]**
buserm[BUSER_WIDTH–1:0][bc]	**-**
bvalidm	**BVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

Read address (AXI-AR) channel signals

Table A-10 lists the AXI read address signals for the AXI master interface.

Table A-10 AXI-AR signals for the AXI master interface

Signal	AMBA equivalent[a]
araddrm[AXI_ADDRESS_MSB:0][b]	**ARADDR[31:0]**
arburstm[1:0]	**ARBURST[1:0]**
arcachem[3:0]	**ARCACHE[3:0]**
aridm[AID_WIDTH–1:0]	**ARID[3:0]**
arlenm[3:0]	**ARLEN[3:0]**
arlockm[1:0]	**ARLOCK[1:0]**
arprotm[2:0]	**ARPROT[2:0]**
arreadym	**ARREADY**
arsizem[2:0]	**ARSIZE[2:0]**
aruserm[ARUSER_WIDTH–1:0][bc]	**-**
arvalidm	**ARVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

Read data (AXI-R) channel signals

Table A-11 lists the AXI read data signals for the AXI master interface.

Table A-11 AXI-R signals for the AXI master interface

Signal	AMBA equivalent[a]
rdatam[AXI_DATA_MSB:0][b]	**RDATA[31:0]**
ridm[AID_WIDTH–1:0][b]	**RID[3:0]**
rlastm	**RLAST**
rreadym	**RREADY**
rrespm[1:0]	**RRESP[3:0]**
ruserm[RUSER_WIDTH–1:0][bc]	**-**
rvalidm	**RVALID**

a. See the *AMBA AXI Protocol v1.0 Specification* for a description of these signals.
b. The bus width is set when you configure the TZASC.
c. The use of this sideband signal is user-defined.

A.3 APB signals

The TZASC provides a single APB slave interface. Table A-12 lists the APB slave interface signals.

Table A-12 APB slave interface signals

Signal	AMBA equivalent[a]
paddr[31:0]	**PADDR**
penable	**PENABLE**
prdata[31:0]	**PRDATA**
pready[b]	**PREADY**
psel	**PSELx**
pslverr[b]	**PSLVERR**
pwdata[31:0]	**PWDATA**
pwrite	**PWRITE**

a. See the relevant AMBA specification for a description of these signals.
b. This signal is not available if the TZASC is configured to support the AMBA 2 APB protocol.

Note

Depending on the configuration of the TZASC, the following section describe the AMBA APB signals:

- *AMBA Specification (Rev 2.0)*
- *AMBA 3 APB Protocol v1.0 Specification*.

A.4 Miscellaneous signals

The following sections describe the miscellaneous signals:

- *secure_boot_lock*
- *Interrupt*.

A.4.1 secure_boot_lock

Table A-13 shows the **secure_boot_lock** signal.

Table A-13 secure_boot_lock signal

Signal	Type	Source	Description
secure_boot_lock	Input	External	When this signal transitions HIGH, it enhances the security of the TZASC by preventing write accesses to certain registers. See *Preventing writes to registers and using secure_boot_lock* on page 2-12 for information.

A.4.2 Interrupt

Table A-14 shows the **tzasc_int** signal.

Table A-14 tzasc_int signal

Signal	Type	Destination	Description
tzasc_int	Output	Interrupt controller	If this signal is HIGH, then the TZASC has denied the AXI master access to a region. **Note** The action Register controls whether the TZASC asserts **tzasc_int** when a region permission failure occurs. See *Action Register* on page 3-8.

Appendix B
Revisions

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

Table B-1 Differences between issue A and issue B

Change	Location	Affects
Modified the block SMC and Flash memory	Figure 1-1 on page 1-2	r0p0
Added Timing Closure options	*Features of the TZASC* on page 1-2	r0p0
Removed figures in Chapter 2 *Functional Description*	Chapter 2 *Functional Description*	r0p0
Added AXI slave interface attributes for Read acceptance capability, and Write acceptance capability	Table 2-1 on page 2-2	r0p0
Modified the AXI master interface attribute values to show that issuing capabilities are configurable	Table 2-2 on page 2-3	r0p0
Added a functional operation block diagram of TZASC	Figure 2-2 on page 2-5	r0p0
Added an example to describe the configuration for subregion disable	*Subregion disable* on page 2-7	r0p0
Combined two sections because both the sections related to region security programming	*Region security permissions* on page 2-8	r0p0
Added an example for Memory map and the register programming	Table 2-5 on page 2-10	r0p0
Removed Tracking Queue description from the book	*Functional operation* on page 2-5	r0p0
Renamed the PrimeCell Identification Register as Component Identification Register	Table 3-1 on page 3-5, and *Component Identification Registers* on page 3-25	r0p0

Table B-1 Differences between issue A and issue B (continued)

Change	Location	Affects
Renamed the Region Attribute Size <n> Register as Region Attributes <n> Register	Table 3-1 on page 3-5, and *Region Attributes <n> Register* on page 3-20	r0p0
Replaced the **apb_clken** signal with **pclken**	Table A-1 on page A-2	r0p0
Added a note about APB signals	*APB signals* on page A-9	r0p0
Removed the table for Scan test	Appendix A *Signal Descriptions*	r0p0

Glossary

This glossary describes some of the terms used in technical documents from ARM.

Advanced eXtensible Interface (AXI)
A bus protocol that supports separate phases for address or control and data, unaligned data transfers using byte strobes, burst-based transactions with only start address issued, separate read and write data channels, issuing multiple outstanding addresses, out-of-order transaction completion, and easy addition of register stages to provide timing closure.

The AXI protocol includes optional extensions for signaling for low-power operation.

Advanced Microcontroller Bus Architecture (AMBA)
The AMBA family of protocol specifications is the ARM open standard for on-chip buses. AMBA provides a strategy for the interconnection and management of the functional blocks that make up a *System-on-Chip* (SoC). Applications include the development of embedded systems with one or more processors or signal processors and multiple peripherals. AMBA defines a common backbone for SoC modules, and therefore complements a reusable design methodology.

Advanced Peripheral Bus (APB)
A bus protocol that is designed for use with ancillary or general-purpose peripherals such as timers, interrupt controllers, UARTs, and I/O ports. It connects to the main system bus through a system-to-peripheral bus bridge that helps reduce system power consumption.

Aligned A data item stored at an address that is divisible by the number of bytes that defines its data size is said to be aligned. Aligned doublewords, words, and halfwords have addresses that are divisible by eight, four, and two respectively. The terms doubleword-aligned, word-aligned, and halfword-aligned therefore stipulate addresses that are divisible by eight, four, and two respectively. An aligned access is one where the address of the access is aligned to the size of an element of the access.

AMBA *See* Advanced Microcontroller Bus Architecture.

APB *See* Advanced Peripheral Bus.

APB Access Port (APB-AP)
An optional component of the DAP that provides an APB interface to a SoC, usually to its main functional buses.

APB-AP *See* APB Access Port.

AXI *See* Advanced eXtensible Interface.

AXI channels, channel order and interfaces
The block diagram shows:

- the order in which AXI channel signals are described
- the master and slave interface conventions for AXI components.

AXI signal names have a one or two letter prefix that denotes the AXI channel as follows:

AW Write address channel.

W Write data channel.

B Write response channel.

AR Read address channel.

R Read data channel.

General descriptions of AXI signals use x to represent this prefix, for example, **xVALID** and **xREADY**.

AXI terminology The following general AXI terms apply to both masters and slaves:

Active read transaction
A transaction for which the read address transfer has been completed, but the last read data transfer has not been completed.

Active transfer
A transfer for which the transmitting interface has asserted the **xVALID** handshake signal, but the receiving interface has not asserted the **xREADY** handshake signal.

Active write transaction
A transaction for which the write address or leading write data transfer has been completed, but the write response has not been completed.

Completed transfer
A transfer for which the handshake using **xVALID** and **xREADY** is complete.

Payload The non-handshake signals in a transfer.

Transaction An entire burst of transfers, comprising an address transfer, one or more data transfers and, for write transactions only, a response transfer.

Transmitting interface
An initiator driving the payload and asserting the relevant **xVALID** signal.

Transfer A single exchange of information. That is, a transfer with a single handshake using **xVALID** and **xREADY**.

The following AXI terms are master interface attributes. To permit system performance optimization, they must be specified for every component with an AXI master interface:

Combined issuing capability

The maximum number of active transactions that the interface can generate. It is specified for master interfaces that use combined storage for active write and read transactions. If not specified you can assume it is equal to the sum of the write and read issuing capabilities.

Read ID capability

The maximum number of different **ARID** values that the interface can generate for all active read transactions at any one time.

Read ID width

The number of bits in the **ARID** bus.

Read issuing capability

The maximum number of active read transactions that the interface can generate. Must be specified if the combined issuing capability is not specified.

Write ID capability

The maximum number of different **AWID** values that the interface can generate for all active write transactions at any one time.

Write ID width

The number of bits in the **AWID** and **WID** buses.

Write interleave capability

The number of active write transactions for which the interface can transmit data. This is counted from the earliest transaction.

Write issuing capability

The maximum number of active write transactions that a master interface can generate. Must be specified if the combined issuing capability is not specified.

The following AXI terms are slave interface attributes. To permit performance optimization, they must be specified for every component with an AXI slave interface:

Combined acceptance capability

The maximum number of active transactions that the interface can accept. It is specified for slave interfaces that use combined storage for active write and read transactions. If not specified then you can assume it is equal to the sum of the write and read acceptance capabilities.

Read acceptance capability

The maximum number of active read transactions that the interface can accept. Must be specified if the combined acceptance capability is not specified.

Read data reordering depth

The number of active read transactions for which the interface can transmit data. This is counted from the earliest transaction.

Write acceptance capability

The maximum number of active write transactions that the interface can accept. Must be specified if the combined acceptance capability is not specified.

Write interleave depth

The number of active write transactions for which the interface can receive data. This is counted from the earliest transaction.

Burst
A group of transfers to consecutive addresses. Because the addresses are consecutive, the device transmitting the data does not have to supply an address for any transfer after the first one. This increases the speed at which the burst occurs. If using an AMBA interface, the transmitting device controls the burst using signals that indicate the length of the burst and how the addresses are incremented.

See also Beat.

Digital Signal Processing (DSP)
A variety of algorithms to process signals that have been sampled and converted to digital form. Saturated arithmetic is often used in such algorithms.

DSP
See Digital Signal Processing.

Endianness
The scheme that determines the order of successive bytes of a data word when it is stored in memory.

Exception
A mechanism to handle a fault or error event. For example, exceptions handle external interrupts and undefined instructions.

JTAG Access Port (JTAG-AP)
An optional component of the DAP that provides debugger access to on-chip scan chains.

Power-on reset
See Cold reset.

Read
Memory operations that have the semantics of a load. See the *ARM Architecture Reference Manual* for more information.

Reserved
Registers and instructions that are reserved are Unpredictable unless otherwise stated. Bit positions described as Reserved are UNK/SBZP.

SBZ
See Should Be Zero.

SBZP
See Should Be Zero or Preserved.

Should Be Zero (SBZ)
Software must write as 0, or all 0s for bit fields. Writing any other value produces Unpredictable results.

Should Be Zero or Preserved (SBZP)
Software must write as 0, or all 0s for a bit field, if the value is being written without having previously been read, or if the register has not been initialized. If the register has previously been read, software must preserve the field value by writing back the value that was read from the same field on the same processor.

Unaligned
An unaligned access is an access where the address of the access is not aligned to the size of an element of the access.

Undefined
Indicates an instruction that generates an Undefined Instruction exception. See the *ARM Architecture Reference Manual* for more information.

UNK
See Unknown.

UNK/SBZP
A field that is Unknown on reads and Should Be Zero or Preserved on writes.

Unknown
An Unknown value does not contain valid data, and can vary from moment to moment, instruction to instruction, and implementation to implementation. An Unknown value must not be a security hole.

UNP *See* Unpredictable.

Unpredictable For a processor means the behavior cannot be relied on. Unpredictable behavior must not represent security holes. Unpredictable behavior must not halt or hang the processor, or any parts of the system.

Unpredictable For an ARM trace macrocell, means that the behavior of the macrocell cannot be relied on. Such conditions have not been validated. When applied to the programming of an event resource, only the output of that event resource is Unpredictable. Unpredictable behavior can affect the behavior of the entire system, because the trace macrocell can cause the processor to enter debug state, and external outputs can be used for other purposes.

Word A 32-bit data item. Words are normally word-aligned in ARM systems.

Word-aligned A data item having a memory address that is divisible by four.

Word-invariant In a word-invariant system, the address of each byte of memory changes when switching between little-endian and big-endian operation, in such a way that the byte with address A in one endianness has address A EOR 3 in the other endianness. As a result, each aligned word of memory always consists of the same four bytes of memory in the same order, regardless of endianness. The change of endianness occurs because of the change to the byte addresses, not because the bytes are rearranged.

The ARM architecture supports word-invariant systems in ARMv3 and later versions. When word-invariant support is selected, the behavior of load or store instructions with unaligned addresses is instruction-specific, and is in general not the expected behavior for an unaligned access. ARM strongly recommends that word-invariant systems use the endianness that produces the required byte addresses at all times, apart possibly from very early in their reset handlers before they have set up the endianness, and that this early part of the reset handler uses only aligned word memory accesses.

See also Byte-invariant.

Write Operations that have the semantics of a store. See the *ARM Architecture Reference Manual* for more information.

Write buffer A block of high-speed memory implemented to optimize stores to main memory.

Write completion The memory system indicates to the processor that a write has been completed at a point in the transaction where the memory system is able to guarantee that the effect of the write is visible to all processors in the system. This is not the case if the write is associated with a memory synchronization primitive, or is to a Device or Strongly Ordered region. In these cases the memory system might only indicate completion of the write when the access has affected the state of the target, unless it is impossible to distinguish between having the effect of the write visible and having the state of target updated.

This stricter requirement for some types of memory ensures that the processor can guarantee that any side-effects of the memory access have taken place. You can use this to prevent the starting of a subsequent operation in the program order until the side-effects are visible.

Write interleave capability
The number of data-active write transactions for which the interface can transmit data. This is counted from the earliest transaction.

Write interleave depth
The number of data-active write transactions for which the interface can receive data.